Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Carla
Gonçalves

Aplicações de IoT no contexto de uma praia inteligente

IoT applications for the smart beach environment

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

Carla Gonçalves

Aplicações de IoT no contexto de uma praia inteligente

IoT applications for the smart beach environment

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Diogo Gomes, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Paulo Barraca, Professor auxiliar convidado do Departamento Eletrónica, Telecomunicações, e Informática da Universidade de Aveiro.

Dedico este trabalho à minha família, que me suportou durante estes incríveis e difíceis 5 anos, aos meus amigos que me apoiaram a manter lucidez e acima de tudo a ser forte a fim de não perder a vontade de acabar este projecto.

o júri / the jury

presidente / president

Professor Doutor André Ventura da Cruz Marnoto Zúquete
Professor Auxiliar, Universidade de Aveiro

vogais / examiners committee

Professor Doutor Óscar Emanuel Chaves Mealha
Professor Associado C/ Agregação da Universidade de Aveiro

Professor Doutor Diogo Nuno Pereira Gomes
Professor Auxiliar da Universidade de Aveiro

agradecimentos / **acknowledgements**

Agradeço toda a ajuda a todos os meus colegas, companheiros, professores, e orientadores.

Palavras Chave iot, cidades inteligentes, aplicações, caso de uso, plataforma aberta, sensores

Resumo

As aplicações de *smart city* estão a crescer em quantidade e qualidade todos os dias à medida que a tecnologia à sua volta melhora exponencialmente. Os meios para criar *smart cities* estão a ficar cada vez mais fáceis de incorporar em aplicações dedicadas à melhoria da qualidade de vida dos cidadãos, bem como organizações institucionais que têm um aglomerado de ideias para expôr em aplicações de uso real. O objetivo desta dissertação é criar um portfolio de aplicação de *smart city* com serviços úteis dedicados ao cidadão comum envolvido no tema da praia, com o uso de dados baseados em sensores. Utilizando casos de uso como a aplicação de sensores de temperatura (vulgarmente denominadas de estações metereológicas), é possível detetar uma variedade de valores relacionados com a temperatura, humidade, pressão, entre outros. Estes, colocados em múltiplos locais ao redor de uma praia possibilitam um estudo mais detalhado e realista de várias áreas geográficas. É também feita uma análise sobre a implementação de uma plataforma *use-case* de um sistema que gere parques de estacionamento - disponibilizando informação online acerca do estado dos parques de estacionamento situados na área balnear da Praia da Barra/Costa Nova (área onde o caso de uso foi aplicado). Junto a estes, é também implementado um estudo a aplicações de pequenos serviços que oferecem informação e suporte a utilizadores que, continuamente, se encontram inseridos no tema e tornam o sistema mais interessante e rico. Conclui-se da aplicação deste sistema, que a informação e funcionalidades expostas tem um impacto positivo e significativo na qualidade de vida das pessoas, não descartando a possibilidade e o potencial de crescimento deste sistema. O objetivo é também permitir que novas organizações e empresas materializem as suas ideias, assim como as que foram feitas nesta dissertação, dando uso aos mesmos recursos - componentes de sensores, fornecedores de dados e pontes de ligação às aplicações.

Keywords smart city, iot, application, use-case, open platform, experimentation, sensors

Abstract

Smart city applications are growing in quantity and quality every day as the technology that drives it improves exponentially. The means to create Smart city environments are getting easier to incorporate in applications dedicated to the improvement of the citizen's quality of life, as well as institutional organizations that have an agglomerate of ideas to expose in real-use applications. The purpose of this dissertation is to showcase examples of a smart city applications with useful services dedicated to the ordinary citizen involved in the beach scenario, with the use of sensor-based data. While implementing use-cases with the aid of temperature sensors (vulgarly named weather stations), it is possible to detect a variety of values related to temperature, humidity, pressure, and others. When placed in various locations surrounding the beach, it allows for a data study with more accurate, realistic results of different geographical areas. An analysis of a use-case implementation of a parking space system is also created- this system ultimately enables an ordinary user access to information, available online, regarding the current state of the individual parking spaces located on the streets nearshore of Praia da Barra/Costa Nova (the designated areas where the use case has been tested). Supporting these, a study of small service applications (that still kept up with the beach scenario) that offer information and support to users, was presently regarded and implemented as a vibrant, interesting additional feature. It's possible to infer from this system's application, that the information and functionalities exposed have a positive, valuable impact regarding the overall citizen's quality of life, without discarding the possibility and potential of the system's growth. The goal is also to allow new organizations and companies to materialize their ideas, much like the ones exposed in this dissertation were.

Contents

Contents i

List of Figures v

List of Tables vii

Glossary ix

1 Introduction 1

1.1 Contextualization . . . 1

1.1.1 What is a Smart City? . . . 2

1.1.1.1 PASMO . . . 3

1.1.1.2 Ílhavo as a Smart City . . . 4

1.1.2 Smart City environments . . . 4

1.1.2.1 Smart City on a technological level . . . 5

1.1.2.2 Smart City on an Institutional level . . . 6

1.1.2.3 Smart City on an Ecological level . . . 6

1.1.2.4 SmartCity as a data manager . . . 8

1.1.2.5 Internet of Things (IoT) applications . . . 8

1.2 Motivation . . . 9

1.3 Goals . . . 10

1.4 Organization of the Dissertation . . . 11

2 State of the Art 13

2.1 Smart city implementations . . . 13

2.2 Platform for Smart Digital development . . . 15

2.2.1 FIWARE . . . 16

2.2.2 SCoT . . . 17

2.3 UI/UX aspect . . . 18

3 Requirements 23

3.1 Applications . . . 23
3.1.1 Parking page . . . 23
3.1.1.1 Stakeholders . . . 25
3.1.2 Beach page . . . 26
3.1.3 Delivery page . . . 27
3.1.3.1 Stakeholders . . . 27
3.1.4 Businesses . . . 30
3.1.4.1 Stakeholders . . . 31
3.2 Real-time communication . . . 32
3.2.1 Socket.io . . . 32
3.2.2 MQTT . . . 33
3.3 User Identification . . . 36
3.3.1 Identiy Provider (IdP) . . . 36
3.3.2 Google Sign-In . . . 36
3.3.2.1 How does Google Sign-in work? . . . 37
3.4 Mapping . . . 37
3.4.1 Leaflet . . . 37
3.5 OpenStreetMap . . . 39

4 Implementation **41**
4.1 System architecture . . . 41
4.2 Server-side . . . 42
4.2.1 Organization of the Server-side folder . . . 42
4.2.2 End-point definition . . . 42
4.2.3 Google Sign-in . . . 43
4.2.4 API connection . . . 44
4.3 Client-side implementation . . . 45
4.3.1 Parking page . . . 46
4.3.2 Beach page . . . 50
4.3.3 Delivery . . . 53
4.3.4 Commerces . . . 56
4.4 Mobile Implementation . . . 58
4.5 System Architecture - Technologies . . . 59

5 Usability Testing **61**
5.1 Cognitive Walkthrough . . . 61
5.1.1 Cognitive Walkthrough Script for Evaluators . . . 62
5.2 Heuristic Evaluation . . . 65

5.3 Usability test results . . . 66
5.3.1 Usability test - Background of participants . . . 66
5.3.2 Cognitive Walkthrough - Setting . . . 67
5.3.3 Cognitive Walkthrough - Results . . . 68
5.3.4 Heuristic Method results . . . 72
5.3.4.1 Visibility of system status . . . 73
5.3.4.2 Match between system and the real world . . . 73
5.3.4.3 User control and freedom . . . 74
5.3.4.4 Consistency and standards . . . 74
5.3.4.5 Error Prevention . . . 75
5.3.4.6 Recognition rather than recall . . . 75
5.3.4.7 Flexibility and efficiency of use . . . 76
5.3.4.8 Aesthetic and minimalist design . . . 76
5.3.4.9 Help users recognize, diagnose, and recover from errors . . . 77
5.3.5 Final Usability test conclusions . . . 79

6 Conclusions **81**
6.1 Outcomes of the developed work . . . 81
6.2 Future work . . . 82
6.2.1 Future work on existent applications . . . 82
6.2.1.1 Parking Page . . . 82
6.2.1.2 Beach Page . . . 83
6.2.1.3 Delivery . . . 84

References **85**

List of Figures

1.1 Plataforma Aberta para o desenvolvimento e experimentação de Soluções para Mobilidade (PASMO) . . . 3
1.2 Internet Coke Machine . . . 9

2.1 Orion Context Broker . . . 16
2.2 UI/UX Implementation Timeline . . . 18
2.3 Initial Draft stage . . . 19
2.4 Mockups of multiple test pages . . . 20
2.5 Live version of the mockups presented above on the previous picture . . . 20

3.1 Example of the application of one sensor per two parking spaces . . . 24
3.2 Google's "Popular Times" feature . . . 25
3.3 Delivery service state diagram . . . 30
3.4 socket.io architecture . . . 32
3.5 MQTT Basic Functioning . . . 34
3.6 Quality of Service (QoS) 2 protocol exchange . . . 35
3.7 Single Sign-On (SSO) working as the access control . . . 36
3.8 Google Sign-In . . . 37
3.9 Leaflet Routing Machine in action . . . 38
3.10 Example of OpenStreetMap (OSM) Map . . . 39

4.1 System Architecture . . . 41
4.2 File Organization of the system . . . 42
4.3 Current live page of the Index page . . . 46
4.4 Current live page of the Parking app . . . 46
4.5 Current state of parking availability . . . 47
4.6 *Go back to car* and *Save car location* buttons on their initial, unchanged state . . . 48
4.7 Parking State Diagram . . . 48
4.8 Leaflet Routing Machine (LRM) being used on the system itself . . . 49
4.9 Current live page of the Beach app . . . 50

4.10 Beach Sensor representation & data . . . 51
4.11 Temperature variation from a period of time . . . 52
4.12 Specific date data retrieval . . . 52
4.13 Current live page of the Delivery app . . . 53
4.14 Delivery State Diagram . . . 53
4.15 Deliverers marked and listen on the Delivery page . . . 54
4.16 Hover action . . . 54
4.17 Active transition signal . . . 55
4.18 Deliverer dashboard: single item of User that has made a request . . . 55
4.19 Current live page of the Commerce app . . . 56
4.20 Live Commerce Page with the detailed information of a single listed business . . . 57
4.21 Mobile Adaptation of the previous pages . . . 58
4.22 System architecture and technologies used . . . 59

5.1 Usability Test Walkthrough State . . . 67
5.2 1.Parking a) results . . . 69
5.3 1.Parking b) results . . . 69
5.4 1.Beach a) results . . . 69
5.5 1.Beach b) results . . . 70
5.6 1.Beach c) results . . . 70
5.7 1.Delivery a) results . . . 70
5.8 1.Delivery b) results . . . 71
5.9 1.Commerce a) results . . . 71
5.10 1.Commerce b) results . . . 71
5.11 1.Parking c) results . . . 72
5.12 Number of scores for *Visibiity of system status* . . . 73
5.13 Number of scores for *Match between system and the real world* . . . 73
5.14 Number of scores for *User control and freedom* . . . 74
5.15 Number of scores for *Consistency and standards* . . . 74
5.16 Number of scores for *Error prevention* . . . 75
5.17 Number of scores for *Recogniton rather than recall* . . . 75
5.18 Number of scores for *Flexibility and efficiency of use* . . . 76
5.19 Number of scores for *Aesthetic and minimalist design* . . . 76
5.20 Number of scores for *Help users recognize, diagnose, and recover from errors* . . . 77
5.21 Heuristic evaluation: Radar chart . . . 78

List of Tables

3.1 MQTT QoS Packets definitions 34

Glossary

API	Application Programming Interface
SPA	Single Page Application
PASMO	Plataforma Aberta para o desenvolvimento e experimentação de Soluções para Mobilidade
IT	Information Technology
IoT	Internet of Things
WSN	Wireless Sensor Networks
MANET	Mobile Ad hoc Networks
QoS	Quality of Service
SotA	State of the Art
IT-UA	Instituto de Telecomunicações - Universidade de Aveiro
LRM	Leaflet Routing Machine
HTTP	Hypertext Transfer Protocol
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
ACK	Acknowledgment
OP	Original Poster
SCoT	Smart Cloud of Things
OSM	OpenStreetMap
JS	JavaScript
UI	User Interface Design
UX	User Experience Design
DOM	Document Object Model
POV	Point-of-View
IdP	Identiy Provider
SSO	Single Sign-On
FTP	File Transfer Protocol
SOA	Service Oriented Architecture
N/A	Non-Applicable
VRP	Vehicular Routing Problem
GDPR	General Data Protection Regulation

CHAPTER 1

Introduction

This chapter ultimately defines what the project will consist of, as well as a set of information regarding all that came in between the conceptualization of the project, and so forth.

1.1 Contextualization

There's always been a great need for human-kind to create the means to improve one's life. This need came in various forms, either by inventing tools that would aid a specific job, to the overall creation of an object that would perform multiple tasks and simplify the process of the user to achieve various goals at the same time.

This is also true technology-wise - creating devices and components that were powerful enough to provide information and made specific actions that seemed otherwise impossible (i.e., communicating with people on the other side of the world instantly, knowing data that wasn't possible to acknowledge for the human senses) can now be considered something as trivial as an everyday physical task. The creation and usage of such devices are also possible due to the increasing number of Information Technology (IT) related people *learning* and *teaching* more, growing in numbers, and performing advanced studies in components and devices as such.

In 2007 when Apple Inc. launched its first model of the iPhone, Steve Jobs (CEO at the time) announced to the world how it was going to be genuinely revolutionized and changed forever: the creation of a device which operated on a multi-touch touchscreen that is nowadays considered to be (along with the following generations), the most used smartphone in the world.

What it was not expected, was that the concept of *smart* devices had now a propeller, or rather, a base on what to evolve from. Smartphones were created as devices that recognized exterior data unseen to the human eye. So, for example, if every smartphone nowadays can capture data with the use of embedded sensors such as temperature values, number of steps (pedometer), GPS location, and others, we are able to also infer that such data can give us information of a more specific type (i.e., if you join the temperature readings with the GPS

readings, you're able to say that the temperature at a determined location is the exact value read by the temperature sensor). What's more is that these components are tiny and sturdy, making it possible to create a small portable device like a phone, capable of doing multiple readings of different sources.

Nowadays studies show that *mobile penetration* is higher than ever (countries like the United Arab Emirates have 80.6% mobile users out of the entirety of the population [1]) and although it is only possible to take conclusions from 2017, compared to previous years which have been showing similar change, the numbers are rising exponentially. Thus, without being aware of it themselves, users are already considering *smart* technologies a staple in their daily life.

The next step would be to evolve from portability and small-task based features to the creation of systems that would be implemented in the actual city and help users. The main idea was to invest in technologies that were able to make a positive impact in the world itself.

The smart city concept came alive, and with them, the avalanche of applications and systems with technology dedicated to the gathering and curing of data for a common goal: to evolve humanity and create systems capable of solving needs to demand, either from the environment, people, and anything in-between.

Much before smart city applications and internet access was available for the users to connect and share data, it was essential to create a base that correctly instigated the development of smart applications. What we are talking about is, of course, the internet itself.

The internet is known to be the most potent tool of transposing information and sharing of knowledge in the entire world. As it is now, it is capable of transferring information and gathering data as fast as the click of a button, which also enables users to gather around and share their knowledge among themselves.

The internet is a helpful platform to showcase the power of smart cities: most of them make use of it as a bridge of communication between users from different points, which also means that smart cities, with this type of aid, are capable of expanding at an alarming rate.

Thus, as the internet made the connection with *smart* applications, these began to have a new title, going by IoT applications.

1.1.1 What is a Smart City?

In this section, we will address the different means to which the *smart city* development applies in the present, past and future times. The main aim is to reaffirm the solutions to which it refers, as well as the exploration of the different cases addressed and open to discussion.

The origin of the smart city concept is one impossible to single-out. There is, however, a variety of systems that have been identified as catalysts and core development, in projects and ideas focused for this purpose.

Due to the accelerated development of today's technology, it is increasingly possible to see practical and sophisticated examples of targeted cities with *smart* technology. The concept *smart* utilized in urban locations was an idea that would have originated in the 1990s, initially as a strategy to expand IT device users in the form of smart communities.

Smart cities are considered to be an urban area that depends heavily on the use of technology-based applications to supply information from sensorial devices and electronic data. The information gathered is often made available to the public knowledge, and it is mostly used to improve, inform, and manage city services such as road traffic, power plants, and others.

The easiest sure-fire way to display this type of information as well as be able to cure it with the aid of exterior, additional data, is through internet communication.

1.1.1.1 PASMO

This dissertation is part of a bigger picture- PASMO.

PASMO (open platform for development and experimentation of solutions for mobility) is a project kickstarted by Instituto de Telecomunicações - Universidade de Aveiro (IT-UA), which in itself is a research facility, composed of multiple aggregations of different universities. Its primary purpose is to invest in research on computers, electronics, networks, and others [2].

While IT-UA is composed of a space where multiple IT people gather to produce numerous technology-enriched projects from years of investigation, the idea to create something more substantial and dedicated to a single purpose came to mind. PASMO was created with that very purpose: to have an open space where people could invest their ideas and follow-up with the actual implementation of these. Most of these ideas are also focused on field-work-implementation. Its primary purpose is to incite research and development for companies that require specific technologies they do not have access to. PASMO gives these organizations a series of equipment as well as knowledge for equipment validation (on the market), protocols, processes, applications, standards, and services [3].

PASMO is, without a doubt, one of the most ambitious projects that came to light. Its single purpose (it is also a non-profit organization that aims to interest people in technology and modern research) is to gather other people as interested in mobility and smart development, as the ones contained within it. Creating future projects without having to forgo their own money- a simple idea, with a win-win mindset for the target audience.

This dissertation is involved with this very project in the sense of implementing a use-case that could be easily expanded using these services. PASMO is composed of three subsystems:

1. **Data Gathering**: From different targets like roads, vehicular resources...
2. **Data Transfer**: Telecommunication and data infrastructure

Figure 1.1: PASMO

3. **Applications**

This dissertation is inserted in the 3rd subsystem.

The point of this project is also to provide complex Application Programming Interface (API)s for whatever company that would decide to put an idea of theirs to use. These APIs would process the data received from the many sensory types of equipment at their disposal and make it *prettyprint*[1] so it can later be connected to the application they decide to put it to use.

1.1.1.2 Ílhavo as a Smart City

When it comes to finding technologies and use case applications for a city that is so invested in tourism like Ílhavo, there were a few ideas that needed to be explored.

For instance, it was essential to explore all the perks the city has to offer- this includes certain things that it is difficult to explore in other places, making it stand out among the different types of existent applications. Be as it may, Ílhavo, in particular, has a lot to offer regarding popular attractions for foreigners and residents included. It is possible to take advantage of multiple smart city types of applications on a single central hub. What's essential in building a smart city idea, is to create its foundation from what the area in specific has best to offer. This can be applied anywhere and is as much essencial to **improve** certain aspects that need some proper solutions in the city, as it is to **create** new solutions that people did not necessarily know they wanted. This is what people are looking for in a successful application, allied to a sharp look that is attractive and easy to use. No matter how complex or simple it is, no one will want to use it if people get stuck with too many things to do and very few ways to do it (i.e., difficulty in traversing your application) or a way too simplified system that offers, virtually, almost nothing of use to the user.

Ílhavo, while having a well-known beach area that is more commonly occupied with a considerate amount of people (varying from seasons), proposes a good point of action to explore the possibilities and the type of data to be retrieved that would revolve around this factor. Furthermore, it is also considered a heavy touristic area - with some local tourist spots and commerce that regularly hosts foreign people, and neighboring local people. This implies that the area itself might be entirely new for most of the citizens that visit it and thus have no knowledge of the services the city provides.

1.1.2 Smart City environments

Smart cities encompass everything that goes from technologies that support the development of a variety of city life aspects; this also includes the involvement in cultural, social, institutional, and environmental aspects. Everything that can be influenced by quotidian situations can be transformed and presented in a *smart* environment.

There's much discussion on what smart devices can offer to a regular user or even a specific city. Many people discuss whether or not this type of applications are invasive or necessary

[1] Prettyprint is the formatting on the stylistic level and syntax in order to expose the information in an easy-to-read format

for everyday living (more recent news regarding user privacy on the internet, which also includes IoT applications, are followed by standards such as the General Data Protection Regulation (GDPR)[4], but it's also been a subject discussed previously in interviews with companies like *Cisco*[5]) as life as it is known has been capable of providing everything a user needs, and thus population tends to be misinformed about the positive influences smart devices can offer. Whether it be directly or indirectly affecting the lives of citizens, smart cities have become almost of a staple to the everyday user (both for regular users of electronic devices, and everyday technology-free users). These projects more than regularly, inconspicuously weight more in the improvement and quality of life.

Most of the use-case applications can be categorized into different, multiple sections. From the wide variety available, four were chosen for argument's sake, as these are deemed the most relevant and influential. Starting from a **Technology** level of applications, **Institutional**, **Ecology**, and **Data Managers**. Although these categories can be subdivided into other sub-categories, they are the most conscious and capable of organizing the very different agglomerate of use-cases that exist to the present day (or even future applications that have been categorized already).

Furthermore, it should be discussed how these four different categories represent on a practical view.

1.1.2.1 Smart City on a technological level

One of the main focuses of smart cities is all the technological development they bring. Research about the production of sensors, software, frameworks, and different types of computing and electronics has brought accelerated growth to the technologies used today to apply these increasingly cumulative use cases. Thus, the number of people in IT has grown considerably, and the demand even more. More and more people are involved in programming and hardware-building elements also to match today's market - this was greatly influenced by the recent surge of ideas surrounding smart city concepts. In addition to the growth of all these technologies, the production of strategies for smart cities has also increased- many examples such as Living Laboratories in Europe, innovation clusters and global hubs such as *Arabianranta, Zaragoza Milla Digital*, and *Seoul Digital Media City* [6].

Intelligent cities with multiple resources to various technologies and frameworks open to the public were made available. Interest has increased, and demand has increased cognitive and learning capabilities. Territories with collaborative spaces, interactive tools, and embedded systems have been expanded and blended as a whole. With this, a new notion of an intelligent urban territory was created.

Because of its ambiguity, the smart city today, in its unique approach, can have a multitude of meanings - both for the development of populational data aggregation technologies and concepts such as ubiquitous (or U-City) cities - that is, interconnected communities using the preferred device of each person. The purpose of ideas like these, for example, is to create a system that has any services anytime a person requires it. This is possible by placing chips in the infrastructures of the community buildings in question [7][8]. The purpose of

structures like these is to enhance the user's reach and knowledge regarding the surrounding infrastructures that make part of their everyday living. To further increase sustainability, automation technologies are increasingly being developed and improved - factories of all types of products already use at least some form of automation systems, which can also be regarded as intelligent solution technologies.

Another principle of technology applied to smart cities was that of Virtual Cities - that is, areas created in a virtualized, non-existent way, that in practice defines a mapping of cabled infrastructures, data centers, among others. An excellent example of an application of such systems is the one in Stockholm. In this city, a smart city project was carried out with a fiber-bonding solution[9]. The company behind this phenomenon, *Stokab*, was founded in 1994. The company itself belongs to the Stockholm City Hall, which makes the project much more involved for co-operative purposes between the population and the government institutions that represent them. This cooperative involvement also leads us to another type of approach to smart cities...

1.1.2.2 Smart City on an Institutional level

In the 1990s (the decade on which smart cities were defined for what they still are in the present day), according to Mary Anne Moser [10], the movement of the *Smart* communities had evolved in a very much exponential way with the primary objective of increasing the number of people involved in IT projects. Together with institutional organizations, the aim was to shape and create environments and projects aimed at improving the quality of life for the population. Government institutions also have felt the pressure of the progression of time, the increasing development of smart devices, and how much time they consumed out of ordinary citizens.

An opportunity to both get the most out of the time spent on devices such as smartphones and laptops (or more succinctly, any form of internet access) and information provided by the user took form. A symbiotic relationship between institutions and the user was created, and organizations were able to offer means and *short-cuts* that consequently improve the quality of the ordinary person's life, just as the population can provide its creative and developmental capacities to these situations in a world aimed at creating technology that would benefit the hefty sum.

Smart city projects also propelled government institutions to increase the technological and overall growth of the city itself. As technology grew to a level where most services that involve finance/banking, city hall services, medical solutions remotely solvable, and even scholar services are a necessity as much as a staple nowadays, Institutions are taking advantage of the new wave of rising *tech* to stand out among other rivaling entities.

1.1.2.3 Smart City on an Ecological level

One of the most attractive smart city solutions revolves around the exploitation of ecologic-centered ideas, with projects ranging from the tracking of energy consumption and natural resources, the premonition of the weather forecast, evaluating pollution levels, and others.

One of the most attractive visions of using and developing smart cities is also the full range of applicable everyday situations. One of these and probably one of the most exploited ones is the one of tracking the energy consumption and natural resources.

Nowadays several cities have adopted technology to detect the amount of garbage in a public bin: ideal to avoid public trash on the ground. There are multiple use cases for this particular situation: In 2011, a company called *Ecube Labs* took a development focused on the creation of the same type of technologies, spread across several countries. One of the case studies of this particular company has also focused on South Korea, where it is already possible to find this type of technology spread across several cities of the country like Bukchon, Seoul, and even universities. These solutions have led to a reduction both in the number of times that garbage collectors did the garbage collection (thus diminishing the cost of garbage collections by a high percentage), as well as a significant increase in recycling, and more importantly, almost permanent elimination of trash overflow [11].

Another popular application to ecological uses is the tracking of energy consumed. Case studies alike *Smart Lighting*. One of the most recent examples of this is the Chicago Intelligent Lighting Program, which aims to install and exchange more than 270,000 public light poles with more efficient, low-cost and dependent LED lights[12]. Programs like these are getting more popular and gathering more attention from municipalities. In addition to obtaining analyzable data, the construction of means and strategies to improve both the quality of life and diminishing the costs of the city makes this type of investment the most sought- and the demand is increasing. Applications such as intelligent lights focused only on users, i.e., a pedestrian/car/bicycle moving in a location covered by this type of technology, that is identified as the agent that triggers the connection of public lights:

1. When the street is empty, the lights are dimmed, thus saving electricity that would otherwise be spent unnecessarily because the area is empty for a certain period.
2. After sensing an object or a moving person, the lights automatically switch on to the maximum potential, also being able of accompanying the moving entity, so that they have the utmost comfort for the entirety of their path.

There are even companies dedicated to the implementation and development of technologies with this concept, one of them is the *Tvilight*, based in the Netherlands, with implemented projects already reporting in Germany, Ireland, and some Dutch cities [13].

Following the same theme of environment-conscious development, new ideas involving the control and tracking of pollution meters to prevent further and find, again, additional tactics to resolve issues surrounding it- knowing the source and where it is most prevalent is one step further solving it. Using various sensors as well as video surveillance connected to IoT technology could effectively mobilize staff for handling risk situations in the environment [14], as well as figure out the cause of most. This kind of project is especially enticing in areas such as Portugal, where in the last few years there have been many incidents, especially of fire-related causes. Most of these go about as natural causes, while some are proven to be

arson - which could easily be prevented and be taken into account more voraciously through systems like the one described here.

Most smart city environment-conscious projects are ultimately the go-to ideas. Due to its life-improving prospects, people in power of investment find this idea the most attractive of all.

1.1.2.4 SmartCity as a data manager

Another idea of smart city usage would be as a data hub: ultimately for gathering information as well as systematic work focused on it, taking a step forth to the means of regional research, as local improvement of risk situations only located by analytic data provided by user information. Data Management comes in 3 different forms [15]:

1. Acquisition
2. Processing
3. Dissemination

All of these have a role in the Data Management spectrum. For data acquisition, the key-point is consistency in data gathering. Although many techniques can be used in different places, the idea is that sensory data from Wireless Sensor Networks (WSN) and Mobile Ad hoc Networks (MANET)s must have the same data format as well as type, to provide for data integration between each other.

The second challenge- Data Processing- is viewed as the evaluation and interpretation of data acquired in the first step. The goal here is to ensure the quality of the data that is to be presented, as well as to evaluate the format, type, and any or which factor that can be of importance so it can be ready for the endpoint. Although each step has some weight on the quality of the data, this step is one where the focus here is most intense.

Data dissemination, however, proves to be one of the most critical steps throughout the whole Data Management process. To provide multiple views of information to the goal users is key to the finished smart city ideal- this, however, is *blocked* by specific difficulties that need to be attended to, such as different types of QoS provided for different users (a.k.a. ordinary user, policeman, owner, etc..).

1.1.2.5 IoT applications

It is essential to address how significant the development of Internet-connected applications is for the basis of a *smart city* environment.

In 1982, it was created what is believed to be the first Internet-connected device- manufactured by a group of students that wanted to be informed if the Coca-Cola machine (see figure 1.2) placed in their department was out of stock (or not). They created an interface that, with three main hardware elements, a Server software, and a finger interface, created a system that made it possible to know in every university computer about the current state of the machine [16]. This story demonstrates the ideal case of how practical ideas (such as identifying the current state of a device/machine at a distance) can be built, as well as how they are growing as an interest to the typical user.

Figure 1.2: Internet Coke Machine

It also showcases a very simplified case of *smart city* applications that focus on helping the typical user: using an IoT application to display information of a given machine/state/environment.

IoT itself is considered to be a system compromised of multiple different physical items such as transports, other electronic devices embedded with varying types of technology (i.e., smartphone), sensors, and connectors that aid the whole system into communicating between each of the devices available.

1.2 Motivation

Seeing as there is a particular lack of applicationa revolving around the smart city concept, this project was built as a demonstrating step on how smart cities could advance. There is a tremendous need for people to start putting ideas forth, especially in countries like Portugal, where there is an avalanche of technology and technology-invested professionals. Seeing as the number of IT people keeps on growing every single day, the advance in technology and finished products should be on-par. This also means that the country (or in a smaller scale- any city) should advance too. It is indeed not because of lack of information or motivation, or even rejection from the general public (who would be less keen in accepting such a rapid advance in technology- or so it would seem). Projects like this can be seen as an advance itself to the overall vision when it comes to possible future smart city implementations, in environments the population wishes to see.

There is, however, already a substantial development of Smart Cities in Portugal. Branching from Porto, where applications using sensory data such as SenseMyCity [17] have been produced making use of about 20 known sensors for a single application (with multiple known spin-offs), or Águeda, a city with over 20 Smart City projects that revolve around the economy, people, governance, mobility, environment, and living [18].

This dissertation, however, is precisely about generating a motivation to work on this type of subject. A system that is equal parts helpful to the society as a whole, as it is for the population. Besides being high on demand, this type of technology is also created as a booster for others to come forth with new, even possibly better utilization for the data gathered by sensory equipment that is also made use of in this project.

Following up the use of this technology, it was also deemed acceptable choosing an area with a lot of population activity seeing as:

1. It remains as a place with a high percentage of users exploring this type of technology (compared with the number of people out of bounds).
2. There's a higher set of data that can be made available to the public due to its high density and market exploration.
3. Seeing as it is an area with low material coverage, sensory devices such as GeoLocation can provide much higher accuracy than the alternative.

Smart City real applications are not only accessible but ultimately useful for improvement of the quality of life, overall satisfaction of the population as a whole, and depending on the case, can prove to be even helpful to governmental institutions. It has shown that it saves money and- at the same time- helps lives.

Mayor Rahm Emanuel, regarding his own city's smart city project, once said[12]:

This project is a win-win – it will deliver one of the largest lighting modernization programs in the country while addressing one of the top reasons residents call 311[2]..

Which, in a way, applies to this very case, and many others around smart cities.

1.3 Goals

The goal of this project is to create a sustainable, informative and highly user-friendly system for people living in the beach area as well as people that intend to go from the outside to the location in specific.

There are some particular necessities taken into account when putting together this project- to create a comfortable, accessible hub that provides all different types of information, as well as raises the quality of life, with the provided data, for the population.

With the creation of this framework, it is also our goal to increase the flow of visitors to this area, if not only for the technology that provides. As well as attracting everyday users, it also remains as a motivator for new smart city framework ideas to appear, hence serving to catalyze possible future implementations.

As it is, this project in specific does not equal to a fully-fledged state-of-the-art application that is open for user experimentation. Rather than that, it was made with a general use-case in mind, providing to other researchers and possible investors in projects like the one where this is inserted in (PASMO) a typical user-friendly dashboard which can evolve to a more significant system that provides a similar structure to the one explored here.

[2]311 is the number used for non-emergency calls, used increasingly in U.S. cities

It is vital to maintain an attractive overall look of this "hub", also focusing in the UI/UX point of the framework, showing this way how such a use-case can be highly-usable and interactive given the right tools and resources for a standard user.

1.4 Organization of the Dissertation

This dissertation shall be organized in the sense of explaining the motifs involved with the creation of this project and also the technologies made available for its development- i.e., SCoT. Tthe State of the Art will be composed mostly of incorporating a small part of history in the technology subject, as well as the different kinds of existing applications for smart cities. Regarding the technology for development, an analysis of the frameworks and tools used, as well as similar ones available nowadays that can offer services akin to the ones approached is provided.

The primary goal is also to provide a variety of solutions that can propel other projects to come to life but without getting fixated on the set of technologies or languages that were exposed in this dissertation.

Since the technological world of Smart City systems is so broad and varied, it is unfeasible to limit oneself to only a few technologies that might not even prove useful in the situation at hand- the goal here is also to explore those options.

Furthermore, as we have passed the end of the State of the Art (SotA), a discussion on the implementation as-it-should-be first (rather than the as-it-is), multiple points of a perfect-to-be implementation that was made impossible by whichever complications it was met with are explored. This will also consequently begin as an introduction of sorts to expose the actual implementation of the project, discussing multiple aspects such as the system Architecture, the dashboard look, languages used, etc.

After the implementation, a thorough survey on the interface's usability was made, as well as a detailed walkthrough regarding Usability Testing in itself. Since the application is meant to be met with a lot of different users, the tests are particularly explicit and show overall consistent results, as we will be able to see.

After closing the implementation and usability testing chapters, the conclusion is reached, where details of some possible future work are made, as well as the final thoughts regarding the project as a whole.

CHAPTER 2

State of the Art

In this chapter, the concepts associated with the technologies used in this project will be discussed. Several techniques involved in the ideas presented here are employed in recent and thematic projects related to smart cities, while also analyzing new possibilities for better improvement and employment of IoT solutions on the subject at hand.
There are several examples covered over the years on this subject - especially in smart cities, which is in an accelerated and expandable development. For the development of these technological projects, implementations of multi-component platforms and solutions for process automation, data maintenance, applications of big data services, and complex information systems will be considered.
For the use of these platforms, it will also be taken into account the application and secure access to these - through authentication protocols (such as OAuth 2.0) and messaging protocols (used in particular for access to data originating from sensors from various sources).

2.1 Smart city implementations

Starting off the exemplification of the use-cases already present in our world regarding smart cities, the criteria chosen to select the discussed applications are: to be city-wide and incorporated in an IoT network of sorts, to be involved in the accelerating development of citizen's quality of life (as most are), and possibly to be within the theme of the one discussed on this Dissertation.

Although some use cases have already been previously named for example's sake, it is important also to discern the main *competition*, or on a brighter note, the main *inspiration* for an implementation of a system like this. For a more world-wide effect, we shall take into consideration:

1. **Smart Santander** [19]: A European-based project that envisions the multiplication of research, resources, and support for applications and services for smart cities. Although European-based, it has a world-wide city-scale spread. It also has the objective of

incorporating horizontal and vertical federation (a.k.a. in the same power-level as higher-leveled organizations) with other experimental facilities that also focus on research and development. Thus stimulating a solid group of investors in technology as well as the creation of multiple new applications for various types of end-users (including, of course, advanced research on IoT technologies). So far it has mobilized the implementation and deployment of multiple smart city infrastructures in various locations, such as the set-up of 20,000 sensors in Belgrade, Guildford, Lübeck and Santander (on which has been deployed 12,000 sensors)- this multitude of components has generated a large variety of technologies and use-case developments.

2. **Barcelona's Smart Mobility**[20]: A project from 2011 which involves multiple technologies for smart parking, smart traffic, and others. The primary goal of this project was to begin a revolution concerning neighborhood standards given to the population.
3. **New York's Smart City plans**: NYC has been a particular point of interest as of late- multiple interesting new technologies have spiked up the interest on developing for this high-tech city. Companies like IBM have created, for instance, a center for advanced analytics, to develop and invest in the creation of infrastructures devoted to public safety, transportation and traffic, water, and energy optimization[21]. More big-name companies have come into the picture for this city: Cisco has also been known to participate in the Smart City movement, especially in New York- cooperating with City 24/7, they intend to create Smart Screens, platforms that are to substitute phone booths, and ultimately provide better security for citizens [22].

Whereas, for a more national standard of systems that have been implemented, the population is turning onto a path of active development of sharp, secure applications that supply helpful, clean information.

1. **Vodafone's Smart Cities:** Vodafone's smart cities plan [23] contains a complete set of features seen to the IT world of smart city implementations. It goes from three different main areas of discovery:
    - **Management and Efficiency:** Where four different areas of interest are investigated. Smart Buildings (for efficient resource management of buildings), Smart Lights (for city-wide lightning efficiency), Smart Waste, and Smart Water.
    - **Tourism and Recreation:** Due to Portugal being a great area of interest for tourism, two applications were considered, that although categorized on this section, it is not too out there to implement them on other types of situations. These are: Smart Counting (which in theory works great for informing outside users of how many people are in a determined location and thus be considered as a factor in deciding whether or not to visit said location), and Smart Mapping which offers more specific, user-friendly information for citizens and tourists alike.
    - **Society and Citizenship:** The final section described, involves the development of Smart Traffic options (whose job is to monitor traffic light information), and Smart Parking which is an implementation that has already been considered by multiple other projects.

The main point of this plan from Vodafone is to provide means to develop these ideas with proper investment in return.

2. **agueda is a SMARTCITY:** Águeda's project of creating multiple smart city solutions [18] to fruition is one that's composed of various different *smart areas*, going from *economy*, *people*, *governance*, *mobility*, *environment*, and *living*. All of these areas are composed within themselves of different applications that identify with the area described. At the moment, it is being propelled and advertised by the city hall, as a way to encourage new technologies and applications to come to life and being showcased in a way that can be reached far more easily by the general public.

Portugal has recently been involved in many projects regarding smart cities - people are becoming more aware and interested in smart applications and smart environments, as these gain popularity not only for their ecologically-friendly prospects, as well as the overall helpfulness it provides to the ordinary user. The current state of evolution stands by the creation of multiple hubs that focus on the creation and development of smart technologies, as well as a high investment in research of such areas.

All over the country, the number of smart research hubs is increasing. Some of the examples in this particular case is the **Lighting Living Lab** from Águeda [24] - whose focus revolves around the development of light-centered smart technologies-, RENER Living Lab which involves around 43 different municipalities all over Portugal - it is currently known as the Portuguese Smart Cities Network, a space for development, testing, and experimentation of smart solutions for urban areas [25] - , Penela's Smart Rural Living Lab - which also focuses on the production of innovative and competitive development [26] - , IT-UA, which can also be identified as a smart research hub, as it offers tools and is composed of multiple developing parties whose focus is, among other things, to create IoT applications that also use sensorial information to create smart applications, and others.

It is essential to grasp that the development of technological projects in urbanized areas is evolving more as times goes. Portugal is one of the many countries that consistently strives to evolve and create more systems that help not only citizens but also the environment. The number of investors wanting to follow-up on the materialization of ideas involved in smart city-centered pieces is also steadily increasing in Portugal, as people soon realize this type of development is, indeed, the future.

2.2 Platform for Smart Digital development

When approaching the subject of using an ideal platform that supports all data coverage as well as the communication between all elements of the application, it is essential to select some crucial factors and chose the best option. Some of these include what type of Database is ideal for the application's usage, the kind of communication the application elements require, how the sensors communicate with the rest of the application, the costs of such platforms, or even how the APIs will be constructed.

There's a variety of platforms available that serve this very purpose: offering technologies and components that are ideal for Smart City platforms. Not only do these offer features that make communication with sensors easily, but they also provide a whole other set of options that eliminate the need to use extra components, as these usually already cover most of everything a Smart City application needs.

2.2.1 FIWARE

FIWARE's Open Source Platform is the closest example of the open platform used as the auxiliary for the implementation of this Dissertation's project.

Fiware disposes of a multitude of services which they call "enablers". They contain multiple open source components, which ultimately serve to assemble into a unique piece of whatever final platform the system would need for the smart application. It is particularly useful for **Big Data** gathering, but also offers a set of complete tools for managing and processing context information [27].

Besides offering many components for data gathering, Fiware provides authorization and access control management for APIs, IoT agents that connect and gather information from sensors efficiently, a central Fiware Context Broker that works as a middleware, Big data analysis (with historical details), platforms to create an application platform, and real-time processing of media streams (such as video, microphone, etc) [28].

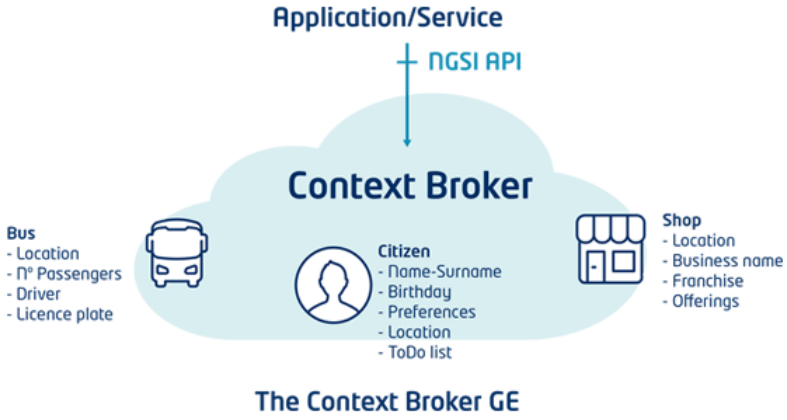

Figure 2.1: Orion Context Broker

Fiware uses a Publish/Subscribe Context Broker for their execution of the middleware and data gathering: the Orion Context Broker. This component is capable of registering context producer applications, update context information, being notified whenever an update is made (or periodically), and query specific details.

The context broker is one of the examples of a component usage from Fiware. What they aim to produce is a complete system using multiple components they offer like the ones described above. This overall creates a wholesome platform that knows how to work for themselves, and how to do it most efficiently.

This type of service is particularly useful in fast-advancing applications that require a complex back-end service, without all the complications of building it from scratch. Besides the fact that this type of service is open-source (i.e., expense free), it is also time-efficient and made in the cleanest way possible. Using Fiware in specific, only the components and models that are needed are selected, without having a whole Server side that offers too much for an application that only requires a particular set of operations.

When using a service like Fiware, and application can be built as elaborate or as simple as it is required, providing the liberty of choice to the owner, and, as a developer, to build a clean, smooth application with strong user experience and usability.

2.2.2 SCoT

There are other, more local designs of a similar concept: the Smart Cloud of Things (SCoT) is one of the other examples of what a platform component-driven can offer to future stakeholders that want the availability of fast-developing systems like these.

SCoT, in particular, brings a set of components that consist of databases that register document-type data, authentication methods, API recognition, and aspects of the network, device management, services, applications, and data mining concepts [29]. It is also capable of producing and aggregating several tenants (which are identified as the deployers of their services) with agility and reduced time to market, making it possible to own multiple services in a single owner format. In this case, it was the one developed with the aid of, serving as a middle-ware system between gathering information from sensor components and connecting to the back-end service of the platform.

Ultimately, it is essential to understand the needs of the system as well as to conciliate those with the needs of its stakeholders. To sacrifice total freedom of implementation is to save time and resources, it can also dictate how robust the system's foundation is and how capable of evolving it is.

When developing systems on top of Open Platforms dedicated for this type of development (such as Smart City systems, which come with a variety of features and use-case implementations), we're also developing an application that will be capable of evolving in a much seamless, more accessible way than one built with a system from scratch. Using platforms like SCoT implies easier connections to sensorial devices, and consequently to the application, as it is more straight-forward than having dedicated connections implemented by the developer.

In an ever-growing city with an almost unlimited gathering of data, it is essential to take into account how capable a system is of gathering said data and make it processable and understandable. SCoT comes as a tool with significant power towards this skill, the gathering of data in an effortless, fast manner, making the system capable of producing more intelligent analyzed content.

As SCoT is built as a Service Oriented Architecture (SOA), it implies that all services connected with it are capable of logically representing business activity with their specified outcome, are self-contained, function as a black box for the consumers[1], and are capable of

[1]Here, black box is presented as a device that has no previous knowledge of the inner workings of the

containing other underlying services [30].

2.3 UI/UX aspect

When it comes to developing functional applications that are visually appealing and attractive for any standard user, one of the most important aspects to take into consideration is the UI/UX.

UI/UX is considered the most crucial aspect of web development- it covers both the visual aspect as well as the functionality of the system as a whole. This paradigm in specific has also suffered much change in the past years.

User Interface Design (UI), as the name might give away, takes care of the Interface aspect of the application. It is the primary focus for the usability needs of the user and takes responsibility for evaluating the best usage a page can give. There are a series of factors to take into consideration- in 1995, Jakob Nielsen published an article citing the "10 Usability Heuristics for User Interface Design" which over 20 years later can still be quoted and followed by everyone developing their system [31]. Among these- although all should be followed- are found some of the most basic and truthful laws to follow such as **"User control and freedom"**, **"Consistency and standards"**, **"Recognition rather than recall"**, **"Aesthetic and minimalist design"**, **"Help users recognize, diagnose, and recover from errors"**, etc.

Many libraries are/have been created to help users better implement these paradigms into their web or applicational service. One of the most public HTML/CSS/JS libraries that cater to these needs is, for example, Bootstrap. It is a fair example to use in this case of use- it aids every user to build responsive websites as its motto goes "Mobile-first" (which means that element sizing and organization prioritizes mobile-view rather than desktop). It is also composed of multiple elements (buttons, headers, page organization) that, although ultimately is of the user's responsibility to organize, they implement an aesthetically pleasing foundation for building web applications.

As a UI/UX developer- whether a designer or software developer- the implementation is usually sectioned off in the following parts (see figure 2.2):

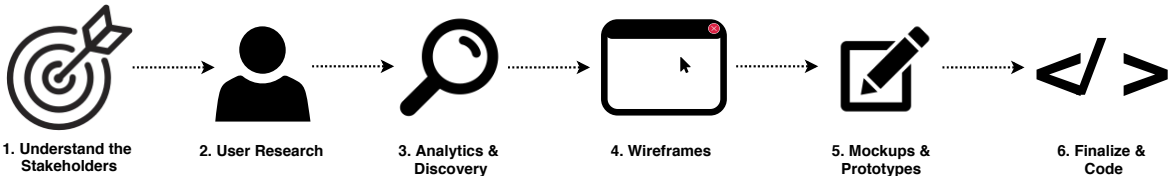

Figure 2.2: UI/UX Implementation Timeline

1. Understand the problem you're trying to solve as well as the stakeholders it is directed to: Depending on the kind of user you're trying to expose the system to, it most probably is going to need a different approach for each of the target audience (or, if universal, it needs to be a generalized system).

system, as it knows only the inputs and the outputs resulting of the running system

2. User research, combined with the first item, is the most reliable source for building and designing the system. As the source would be directly telling the stakeholder what their needs are as well as inform them of any new features they might not have considered previously.
3. Analytics takes a big slice of the importance of application design: It helps the stakeholder to understand what elements are easily reachable and most importantly, what elements are not. This also includes information that is reachable to the users and the overall analysis of the system that is designed to the user.
4. Conceptualizing the application is the first step toward envisioning the final product: The key features are placed, as well as organized accordingly to the previous research that has been done. This stage expects small drafts of what the final project is capable of being (see figure 2.3).

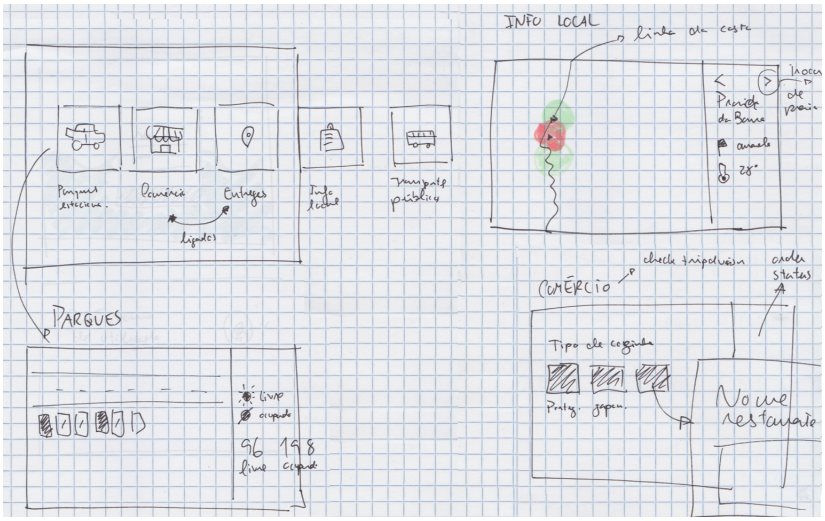

Figure 2.3: Initial Draft stage

5. Mockups and prototypes should be presented as a more refined version of the Wireframe stage. No longer is the system being imagined, it is being produced. Every aspect in here must be approached, as it will extend to the final implementation of the application (although a few adjustments can be made on the developing phase, it should not deviate much from the ones presented in the previous step). Some people also choose to define the interactions and develop more ambitious-looking prototypes of the application. There are a few applications that serve this very purpose, although they are very Design-team centered (i.e., products produced by Adobe, for example, like Adobe After Effects recreate the interactions and the motions of the movements of the page in animation-form, Adobe Photoshop is capable of recreating page elements with total freedom, or simple online websites offer similar, free, more limited products that are usually of interest to software-focused developers) (see figure 2.4 for the example of the evolution from the Wireframe phase).

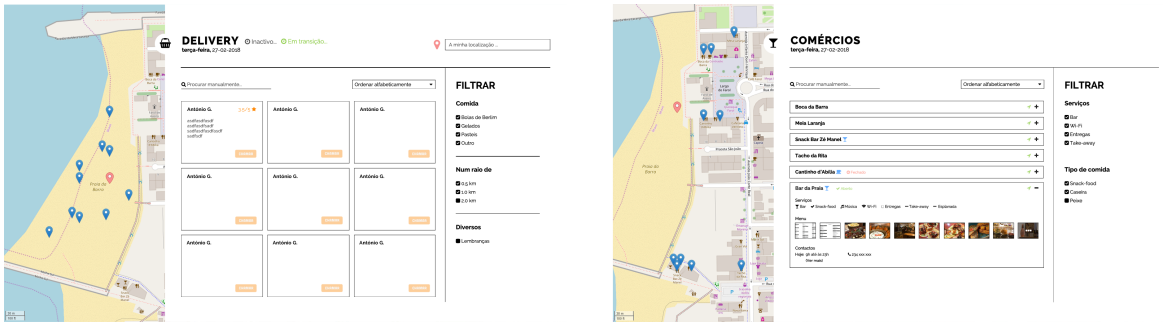

Figure 2.4: Mockups of multiple test pages

6. Finally, the last step of UI/UX implementation is the actual implementation of the code and finalization of the project. All components designed meticulously in the previous step have to be incorporated in live-form, such that the user experience can live up to the user interface that has been so far produced. Although it is the last and only step of this timeline, it is one of the most time-consuming and hardworking stages of the program.

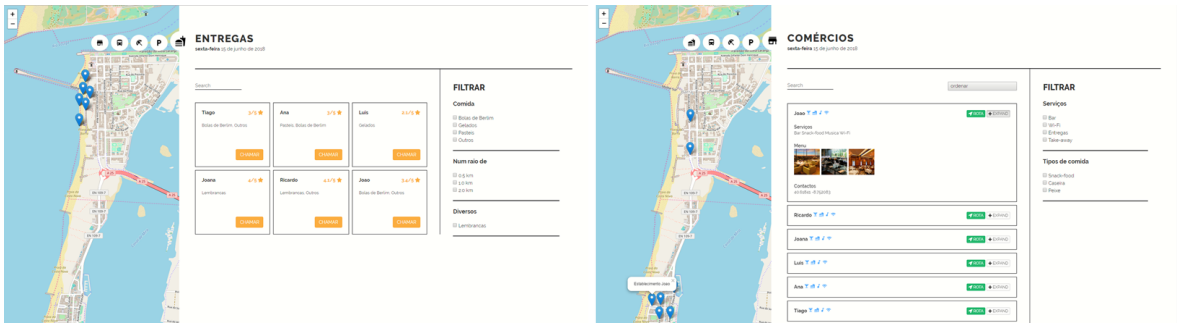

Figure 2.5: Live version of the mockups presented above on the previous picture

While UI represents the overall look and feel of different pages (which in itself dictates how the user, through "the power of sight" will have difficulty- or not- in accessing some aspects of the application), User Experience Design (UX) focus on the experience of the user as a whole.

It is a more complicated process than UI, as it is composed of multiple elements going from improving user satisfaction, ease of use, and overall pleasure of usage for the user on the interactions with the platform. Since UX is so vital in the development of a product (as it may influence a single user's experience, but also everyone related to that user who can or not recommend using the application), many models have been proposed over the last two decades to define the ideal standard of optimal user experience- this goes from Patrick W.Jordan's "User experience must feel **enjoyable**" [32], to Sari Kujala's "User experience can only be truly evaluated in a long-term usage and with a number of factors to take into account of what truly defines a well-rounded user experience" [33].

One of the most relevant and dramatic changes concerning UI/UX development came with the appearance of the Material Design developed by Google[34]. Material Design is a design language that was built in 2014; it is regarded as one of the most substantial advances concerning allied user experience with design. Although it is a recent development,

its simplistic and easy to read layout-aspect is, nowadays, the standard used by all of Google's applications (including the overall aspect of their web applications as well as mobile).

It is safe to compare Material Design to a paper-like layout, using grids, plans, and icons that reminisce card-like browsing.

It is relevant to talk about Material Design on this subject in particular as it ultimately inspired some of the outlooks for the project: the content boxes, the responsive mobile look, etc. Google also made available the icons that were developed and designed for Material Design which, on this project, were the ones used.

CHAPTER 3

Requirements

After the initial plan for the development of this use case, four ideas were materialized to be implemented in a single serving hub-like application. All of these will be displayed as a Single Page Application (SPA), meaning that the interface's look will be standardized and transitions between pages will be seamless when it comes to interchanging pages.
Each of these will feature, uniformly, a white sidebar situated on the right side of the screen (on a web visualization no smaller than 1024 pixels of width), as well as a map occupying up the remaining space, situated under the sidebar. The sidebar will display different data on different pages, as well as feature an option to navigate between them.
For the visualization of this type of data, a back-end is needed, responsible for rendering , that will render the look for each page as well as connecting to the APIs that serve the correct interfaces with the data they are supposed to show.
For server interaction, all the data that is connected with the SCoT (a.k.a. sensor data), will be served to the server through a MQTT connection, and consequently served to the client through a technology called socket.io.

3.1 Applications

To better understand the requirements of the system as a whole, each application will be divided and discussed according to their needs and optimal implementation.

3.1.1 Parking page

There are multiple parking spaces available to the user. These spaces have two possible status: occupied and vacant. All of the spaces marked in the map will be vacant, as it would be pointless to show occupied spots that wouldn't be available for the user. The information will come from different sensors, meaning that each parking space will be connected a sensor that magnetically detects if a car is over it or not.

There were some reflections regarding saving resources and money regarding the sensors used for this page:

- Sensors could be positioned in **each specific place** (which, in retrospective would be the most resource-heavy choice, but the most accurate) and placed in a way that would be impossible the immobilization of these (i.e., theft).
- Each sensor would **cover the minimum of two spaces**, detecting free spaces but disabling the user of knowing if there are one or two available places for the taking as well as what space of the two covered by the sensor is the one vacant (see figure 3.1 for reference). On the other hand, it uses up fewer resources and is more cost-efficient (this has been applied mostly to shopping malls as it uses up electricity for it to work).

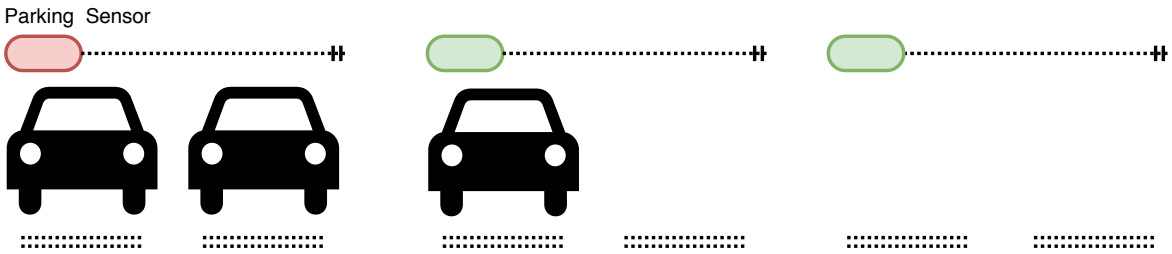

Figure 3.1: Example of the application of one sensor per two parking spaces

- Each sensor would **cover a sequence of parking spaces** of parking spaces, which is the same concept as the one on the previous item (and figure 3.1), except instead of covering two parking spaces it covers a set of spaces that are extended by the sensor. This case is less resource-heavy, but far less precise.

Each of these cases depends on the location of each parking area, regardless of their availability. On the last item described (less wasteful), it would be adequate to use polygon coordinates of the covered area. In more specific parking cases with a higher granularity, like the first one described (each spot takes a sensor), the coordinates describe the exact location of each space.

Another factor to take into consideration is the functionality of the dashboard as a whole. It is important that it not only informs a person of the location of available parking spots, but also for it to be appealing and functional. Since the system already has a straight-forward solution, after a population study, it was unanimous that the most common feature users would rely on is the option to save the car location after it would be parked. Since there's a possibility that the covered area of the parking sensors will be in a semi-large scale, this tool can be useful for the user to accurately access the location where they left the car. This would be achievable by selecting a button with the option to "Save Car Location" . After the parked car location is saved, the option to go 'Back to Car' would be available, in order for the user to come back to the saved space. The "Back To Car" button, would be able to access the location of the user (provided they enabled access to their location by the application) and calculate the route to the location that was previously saved.

This type of mechanic is very accurate due in part to the location where this technology will take effect: the beach. The more unhindered by shadows or buildings a place is, the stronger will the signal of the GPS location be.

There are specific aspects that can facilitate the visualization on the map of these locations: using an adequate skin for the map would also have to be considered. In this case in specific, a study was performed on using the OSM open-source map, which is composed of full detailing, is visually appealing, and free to use (see OSM Section 3.5).

3.1.1.1 Stakeholders

There is only one possible stakeholder for this dashboard: the universal common user. Due to this application only requiring one point of interest, the rest is handled externally, such as:

- Removing, adding, or updating certain elements such as the locations of parking spaces, will be handled Server-side.
- There are two possible routing situations:
  - Finding a free parking space (Car -> Park)
  - Finding the parking space where the user has left their vehicle (Client -> Car)

The main requirements for the user are:

- Observe on the map the available parking spaces.
- Indicate the closest parking space according to the user's location, as well as calculate the fastest way and the corresponding routes the user must take.
- Real-time updates of the current availability state of the parking.
- Save the parked car's location.
- Assisting the user on returning to its car.

Fundamentally, this would be the required objectives for the application, for a stronger application with more user features, it would be interesting to build a system that collects all the daily parking information and would provide a proper study regarding the occupation relative to the hours, days, season, and others. This is a feature based on Google's system of evaluating the busiest hours of a given location (i.e., store, restaurant, etc.). As seen in figure 3.2, according to previous data gathered, it is possible to evaluate the number of users on the location between 2 pm to 8 pm.

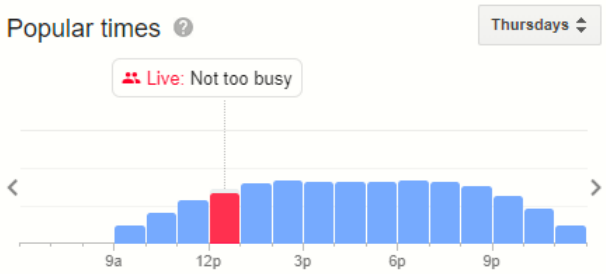

Figure 3.2: Google's "Popular Times" feature

This would help the user to plan their trips in advance, according to the availability of the parking spaces that are offered to them.

3.1.2 Beach page

Given that the chosen area was the beach, there's an extensive list of use cases when it comes to the information it is possible to be retrieved for the user and based on the users. There are some studies relating to the beach that have been put to practice, given that it is already a sure-proof formula for success.

One of the most popular and complete examples is Vodafone's "Praia em Directo", which is available on browser [35], and on the App Store [36].

It is a simple concept- multiple sensors detect a multitude of weather-specific data given periodically and accurately to the interested user. Vodafone's example extends to as many beaches as they could, with a long list of well-known beaches across Portugal's shore.

Although it was a good use case for this type of applications, Vodafone's is mostly outdated, with an uninviting look (it was launched circa 2011), yet it still applies an excellent example of data visualization to follow. Given this, the Beach dashboard (or "Smart Beach") would have a very similar concept in regards to the informative factor and the type of data retrieved.

The sensors needed for this go from:

- Temperature and Atmospheric activity sensor
- Crowd density sensor - This could be provided from the number of connected "accounts", or online users. This also follows the same example as the example in figure 3.2.

The most important values to retrieve from these sensors are:

- Water temperature
- Air temperature
- Air humidity
- Wind speed
- Wind direction
- UV index
- Atmospheric pressure

There's also an interest in adding features such as the graphical visualization of the variation of the gathered data. This would be achievable by selecting the period the user was interested in, or selecting default periods such as "7 days ago", "1 month ago", etc. Some of the values are more important than others, so those would require particular attention when it comes to UI/UX.

Furthermore, using a tool to detect the crowd density would also be useful to the user, not only to know which of the areas would be more crowded but also to evaluate the best place to be at and consequently to park their car. In this manner, one dashboard would influence the other (in regards to the Beach dashboard and the Parking dashboard).

3.1.3 Delivery page

The ideal way to describe this case, is by describing an use-case example applied to a real life situation:

In an occasion a typical user is frequenting the beach, they too have access to a varied list of vendors offering different types ofof products that include food, soft drinks, memorabilia, etc. The usual problem with wishing to buy this kind of products is that most of the time the vendors cannot reach a good percentage of people that are looking to buy them. This leads to a wide discrepancy both in buying and selling products that could be easily solved.

The problem also lies on the difficulty of maneuver for the consumers- what this means is, since consumers are usually situated on their spot on the beach, it is difficult (or not doable) to abandon their belongings (even if for a few minutes) to have either the opportunity of encountering a seller nearby, or having to look for one around the area. The best alternative to this would be to force the consumer to pack their things and reach for the vendors safely, or depend on someone else to look after their belongings. Nevertheless, this strategy is all-together unpractical and bothersome.

The idea is to ultimately facilitate the way vendors can reach the consumers. Thus, the already on-the-move vendors would go to the places where consumers would be interested in buying their goods,and also inform the consumer of where each of the vendors would be and would be, enabling them to reach the vendors more efficiently.

Thus creating a system that detects the location of the vendors of consumer goods on the beach, facilitates the searching of these, and supports the action of requesting a vendor to go to a specific consumer.

On the vendor's dashboard, a list of one or more consumers that have requested him would be available, enabling them to quickly cross the way to the users requesting him as their location would be provided by the application the moment the request was made. The user would also be able to know where the vendor is, as well as an estimate of the waiting time (which depends on the number of people the vendor is still attending to and how far he is from them).

The transaction would not be embedded in the system, as these vendors are already used to performing payments at the moment of transaction.

A method to evaluate the order is also an idea to implement in order to increase the popularity of certain sellers- this would motivate the commercialization to grow and have a stronger service experience provided for the consumer. Having a method that accepts written reviews of regular users would make the process too extended for a for an ordering of simple consumer goods, but a rating system is a simple, effective way to evaluate the service quality of a seller.

With a rating system, it pushes sellers to do a good job to keep the costumers satisfied, seeing as having a higher rating would increase the number of people requesting for the highest rated vendor.

3.1.3.1 Stakeholders

Following all the specifications of the system that is being built, two stakeholders exist in this use-case:

- **Consumer:** The role of the consumer in this situation is to be able to filter the list of vendors from a series of variables in a way that they can find the vendor that is most adequate to what they are looking for. The platform, therefore, will have the following features:
  - **Filter:** The user should be able to filter through the list of vendors with significant variables. A few examples of these variables are Sellers in a specific radius (i.e., within 500m, 1km, 1.5km), Type of Consumer goods (i.e., Food, Soda, Memorabilia), etc.
  - **Ordering:** To organize the list of vendors, there should be a possibility of ordering alphabetically, by rating, by consumer goods, proximity, cost, recently added, etc.
  - **Search:** If the user is looking for some seller in specific (they could have bought from a certain vendor more than once), there's also the possibility of searching the vendor by their name on the platform.

  The consumer should also be provided with a notification system that alerts him in case his order has suffered alterations such as:

  - The request is accepted/refused.
  - The vendor requested by the user is close to their location.
  - The transaction has been accepted/completed.

  The main dashboard used by the consumer will have the following components:

  - The main map that marks all the locations of all the vendors in the available area and the user's proximity. Those markers will be interactable and presented in a live manner, showing up the location of the vendors updated periodically.
  - A list of all the filter possibilities, ordering, and search bar for easier access to a proper list of vendors the user, with their selected criteria, is likely to be more interested in.
    * Each listed Vendor has a small box there's a small description detailing what they sell. Exposing the price of their products is optional.
    * Each of these listed vendors has a button that is used by the consumer to request their services- "Call Vendor". The sellers then receive the request and are sent the location of the person, right after they are inquired on whether or not they wish to go forth with the request (some factors such as too many clients, low stock of consumer goods, etc., can hinder the vendor to accept new requests).
    * After clicking the button used for requesting a service, the user is presented with two different means of verification:
      · A pop-up to verify if the user wants to confirm the request: "Are you sure you wish to call for X?" (in which X is the name of the vendor).
      · The "Call Vendor" button, used to call the service, transforms into a cancel button, if the user wishes to cancel the request that has been made.

- Each vendor is given a rating by the person they sold to at the end of the transaction (see figure 3.3).

- **Vendor:** The vendor's role and their dedicated dashboard are set to expose the product that they currently sell in their text box, the price of their products (optional), and any other point of interest that might engage consumers more easily. It is advised to use a description, to gather a higher level of trust from the consumer who is looking for someone who offers services similar to the ones a given seller offers. A higher level of trust implies that the vendor can more easily get chosen by users, and therefore gain more ratings.

  Ratings, as they usually work on other applications with similar concepts like this one, usually dictate if a seller gets chosen or not, thus the more ratings sellers get, the higher the chances are that they can have a more successful business.

  The vendor's dashboard is a little more simplified than the consumer's. In here, a similar style of layout with a map incorporated and showing the location of the users that have an on-going transaction with the vendor would be available. On the central dashboard, a list of those same users would be available, as well as ways to more easily reach them on the map: this would be achievable with functioning buttons that can redirect the location of the vendor to whatever button with the name "Route" from whatever client he decides to go. The default would be the first client that the vendor got, but it is interchangeable due in part to the fact that some other users that have also requested him may be closer than others (or even, in some cases, on the way to the next customer). A cancel button would also be available for the vendor, as some setback might occur and hinder the seller from concluding a transaction.

  The platform also does not contain any actual transaction component, as it is uniquely and entirely focused on aiding the common user. All the actions performed by the consumer are entirely out of reach for the vendor, and constantly subject to alterations based on the actions of the consumer (influence on the "popularity", or "rating" which is measured by each assisted consumer). The seller has to perform a given set of actions according to the state it is currently on:

  - The vendor may or may not refuse to assist a client that may be requesting him if he is proven unable to perform the given task.
  - After the transaction is complete, the vendor must confirm on their dashboard that the process has been concluded, automatically assuming that a message must appear on the consumer dashboard so that the transaction - and consequently the vendor- may be rated by the client according to their satisfaction (the rating is evaluated from 1 to 5, 5 being the highest).

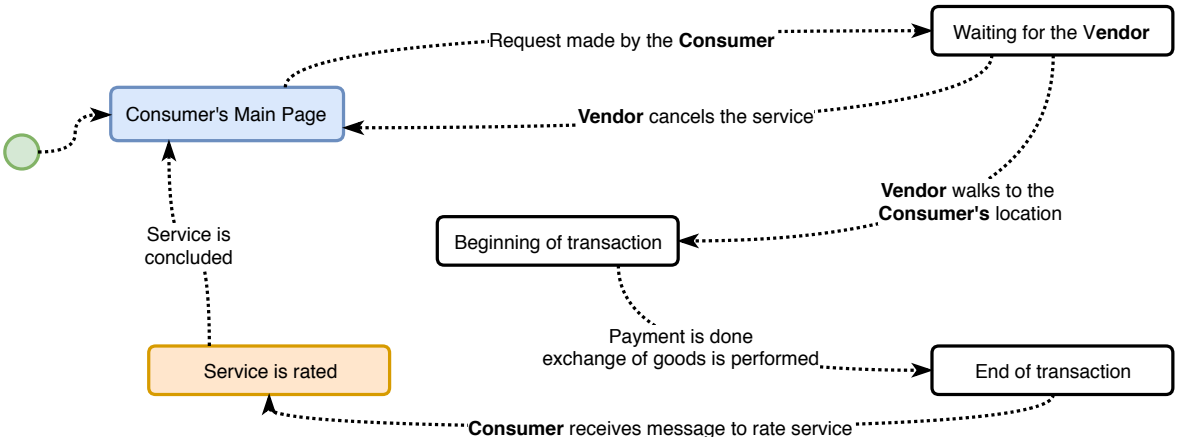

Figure 3.3: Delivery service state diagram

Transaction identifies as the act of trading a consumer good (product) for the payment. The service, on the other hand, includes the whole process from the start (requesting a specific vendor) to finish (rating the vendor).

3.1.4 Businesses

When it comes to raising population awareness of local businesses, there are a few solutions already being explored: from governmental institutions broadcasting local commerces on public venues, to a multitude of existent applications that at the distance of a search for their town, they list all the places nearby the area searched by the user. These are, for example, Yelp, TripAdvisor, Zomato, etc. While these applications all follow a similar formula, with detailed descriptions of the location someone might've clicked on or been interested in (this includes menus, photos of the location, price table, etc.), reviews and ratings done by other users, and overall a simple to use, complete application.

All of these have proven to significantly facilitate the discovery and consequently the number of customers for commerces, all at the distance of a click. This, of course, has proven to be an exciting application to implement- not only to attract outsiders but to increase the overall movement of economy generated from population to business owners locally.

This dashboard is ultimately dedicated to broadcasting the multiple local businesses (near the beach)- this includes, for the most part, bars and restaurants.

After a population report on the primary interests for people from ages 15-50, it was determined that most of the young faction of the people find that it would be most useful to know the information of the bars surrounding the beach, as it's not always apparent what times they're open and is mostly occupied by people from 15-30 years old. When it comes to older people, regarding this business ideal, they believed it is most helpful to know the best places to eat according to price, type of food, and location. It is safe to assume that most recreational types of business are what the majority of the population is looking for, as well as easy access to these.

Making the access to these establishments more natural also raises their value, as more customers can start to actively and more efficiently visit new places (or even old ones that

still want to insert in the platform).

All of these ideas were put together and strategically placed to effectively be showcased in the best way possible in an online platform. Each establishment that is located near the beach (or the indicated area of use for the application as a whole), would be identified as a presumable target to be listed on this dashboard, with information provided by the owner. All the basic information would still be available and mandatory, elements such as:

- **Localization:** Given by a set of coordinates and, as the establishment is listed on their own content "box", with an easy-access button so the user can know how to reach the location optimally.
- **Working hours:** One of the highest factoring elements of interest for common users- to know if an establishment will be open at specific hours.

3.1.4.1 Stakeholders

There are two different possible views for a system design like this: A dedicated page to the owners of each establishment (so they can do whatever they please concerning business descriptions, photos, and such), and the one used for the typical user.

- **Client:** The goal of this service is very similar to the one available for the Deliveries 3.1.3 page, the difference being that this one exists purely for informational purposes. Any direct interaction involving the user and the business establishment is purely on informing basis- this means that there is no "ordering" option, and the features available are of search, filtering (which are similar to the Deliveries 3.1.3), although it still provides a rating method from the user.

  The user is still capable of searching a specific establishment by variables like the name, localization, etc. The listing of each establishment is similar to the Deliveries 3.1.3 page.
- **Business Owners:** Regarding this page, some UI/UX will be optimized for the business owner view. In this dedicated dashboard, the owner of the establishment in question will be able to edit the information regarding his business, as well as define what icons to show to make it more visible (icons to reference the type of establishment it is, for example: (disco-bar: music icon; cocktail-bar: beverage icon; food: cutlery icon)). All of these small details will ultimately turn the establishment more easily accessible and therefore more attractive. The information can also be managed locally (optionally).

  The information editable for the business owner goes from:

  - Name of the establishment;
  - Localization;
  - Working Hours (including any schedule variation it presents);
  - Location pictures;
  - Priced Menus (if existent);
  - Other information represented in icon format (as discussed previously 3.1.4.1).

The user can also rate each business if he wishes to do so, regardless of the location has been visited by the user or not. These ratings will dictate the popularity factor of the area.

3.2 Real-time communication

For this project, it was necessary to implement a form of server/client interaction that would be able to transmit data in real-time as smoothly as possible.

3.2.1 Socket.io

Socket.io is a powerful tool that can send data in real-time from back-end to front-end, or vice-versa due to it being bi-directional. It is a tool mostly used for live interactions, such as online chats, instant messaging, and on this case, direct communication and alterations from sensors/Message Queuing Telemetry Transport (MQTT) messages.

As the name implies, it uses a "socket" implementation, meaning that it establishes endpoints of receiving or sending information.

Its architecture is composed of a Node.js server, and a Node.js client working as javascript framework for the browser.

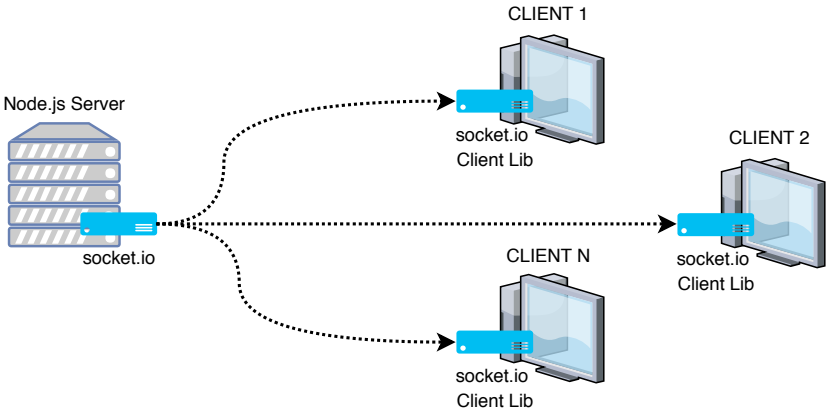

Figure 3.4: socket.io architecture

It also works as a streaming platform for binary data such as video, music, or image: this includes streaming of ArrayBuffer objects, Blob objects (which are file-like objects of immutable, raw data - transmitted from the HTML), and Buffer classes (from Node.js). A lot of different data can be transmitted bi-directionally, meaning that socket.io is a complete multi-type framework.

Socket.io enables sending data from the Server to the Client, providing fast information as soon as it is received from MQTT.

```
@app.route('/mqtt_message', methods=['POST'])
def message():
    msg = request.get_json()
    emit("parking", json.dumps(msg), namespace="/", broadcast=True)
```

Código 1: Example of Server-side emit of information using socket.io

It is a relatively simple tool to use. In this case, it has been defined to use the Broadcast usage since it needs to send all common information to all users. This example is used for the Parking page, and the free allocations will have to be available for all users. It is also possible to define particular views for different users, depending on their identification or their "event name" (in this case, the event name is still present and identified as "parking" (see code 1)).

On the client side, it is presented with the same event name, as well as manipulating the receiving information from the Server-side.

```
function getParkings(my_location){
    parking_socket.on('parking', function(msg){
        var result = JSON.parse(msg);
        setLocations(result, my_location);
    });
}
```

Código 2: Example of the receiving end on Client-side from socket.io Server-side

On the figure above it is possible to infer how even more simplified the whole process is designed. Since socket.io makes the process work as a direct connection after opening a session. The received information is gathered by having the same event name (the connection identifier), working as a Publisher/Subscriber and using its callback as the data received.

This data is received as an object and transformed into a JSON-format type. On setLocations(), the data is processed and positioned in the correct scheme of the front-end side.

3.2.2 MQTT

MQTT is a publish/subscribe messaging protocol, used primarily in systems that require a small *code footprint*, where network bandwidth is limited or is of high-latency. It is designed as a very lightweight protocol, ideal for usage in constrained systems/devices that have very low-operating power/batteries and cannot function with an otherwise massive messaging system. The usage of this protocol also implies the creation of a message broker, a middle-man module that translates the message going from the sender to the receiver- it performs tasks such as translating, validating, and routing [37].

MQTT itself ensures reliability and some level of assurance of delivery. It is ideally used in Machine-to-Machine (M2M) systems or IoT. Mobile applications are, for example, one of the prime examples of best usage of MQTT [38].

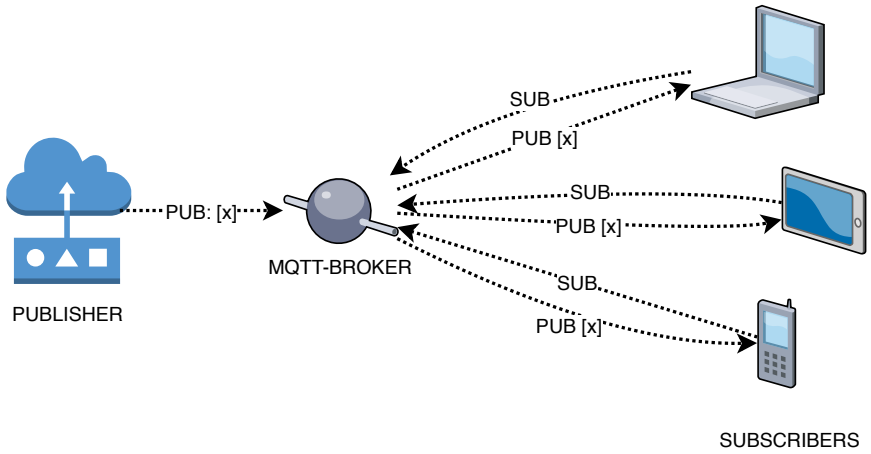

Figure 3.5: MQTT Basic Functioning

MQTT also disposes of a series of control packets to ensure full functionality and control over the messages it sends, these are divided in packets sent by the Client and by the Server [39]:

- **CONNECT**

  The Connect packet is sent by the client to initialize a new connection with the server. Were the client to send a second packet, the protocol assumes it as a violation and disconnects the client. This packet is, therefore, sent one single time the instant the connection between client-server is set.
- **CONNACK**

  A Connack packet is the first message sent from a Server to a Client: it works as an acknowledgment of the connection the client is setting up, and it is a response to the **first** Connect packet the Server receives.
- **PUBLISH**

  Either Client or Server are able to send a PUBLISH message: the client sends an Application Message to the server in order to the server be able to distribute to other clients with matching subscription to the original sender, whereas the server sends an application message to the client whenever the client has a matching subscription to the one on the Original Poster (OP). QoS can assume three different levels:

QoS value	Description
0	At most once delivery
1	At least once delivery
2	Exactly once delivery

Table 3.1: MQTT QoS Packets definitions

- **PUBACK**

  An Acknowledgment (ACK) of the PUBLISH message.
- **PUBREC, PUBREL, PUBCOMP**

  Part of the QoS 2 protocol exchange: see figure 3.6.

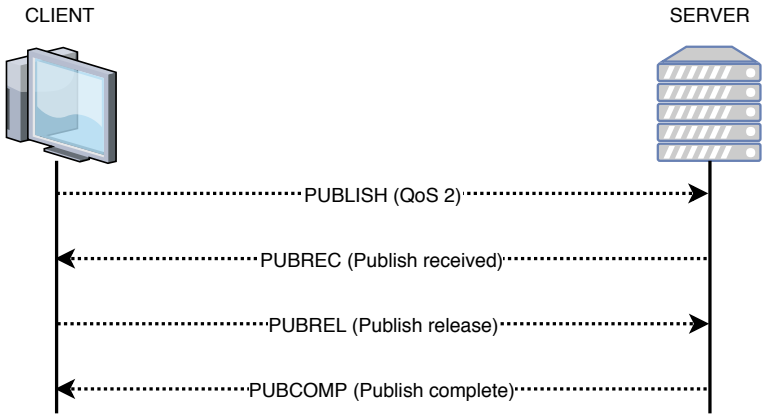

Figure 3.6: QoS 2 protocol exchange

- **SUBSCRIBE**

  The SUBSCRIBE packet is sent from the client to the server to subscribe to a specific topic. This enables the client to receive all the information regarding the topic it subscribed to as the server PUBLISHes it. The SUBSCRIBE packet also specifies for each Subscription the maximum QoS the Server can send Application Messages to the Client.
- **SUBACK**

  The SUBACK message is sent from the Server to the Client to verify that a Subscription has been done. It is mandatory, as it is sent with the identifier of the Subscription it pertains to.
- **UNSUBSCRIBE**

  Whenever a Client wishes to remove themselves from the subscribing list of a specific topic, it sends an UNSUBSCRIBE packet.
- **UNSUBACK**

  ... which is followed by an UNSUBACK, as the confirmation receipt.
- **PINGREQ, PINGRESP**

  The PINGREQ package is used for multiple functions, but mainly for a maintenance purpose for the server or client. It can indicate that the Client is "still alive", if they have not sent any request in some time, or sent as a request to the Server for it to confirm it is still running. Overall, it is used as a Ping request to verify network connectivity between one or the other. The PINGRESP serves as a reply to every PINGREQ sent, either from Client or Server.
- **DISCONNECT**

  The DISCONNECT packet is used by the Client to be sent to the Server to verify that the other side has disconnected in a clean matter.

MQTT is an excellent tool for working in Smart Systems, where the use of sensors that have a constant flow of data being produced is a given. Where MQTT comes here, is the simple act of forwarding all that data into the given endpoints in a fast, intuitive manner,

informing the Subscribers of their subscribing topic of the data being received instantly- since MQTT works smoothly with sending/receiving data.

MQTT, in this project, consists of a Server-side broker that forwards the data to its needed places, and each API subscribes to a different topic set up on SCoT. The API then forwards the data through socket.io much like the example in code 1.

3.3 User Identification

As much as this platform is merely informative, it offers some functionalities that require user identification. Some features from the Parking dashboard 3.1.1, the Deliveries dashboard 3.1.3, and maybe future features that may be implemented need to differentiate between different users.

For this, the best possible implementation for a subject as such- with a number of applications all to be used by the same user on the same device must be taken into account. Considering an approach such as an IdP is adequate.

3.3.1 IdP

An IdP is a system that mostly creates and manages the identity information of a specific party that has signed into a providing page. It offers user authentication as a service. Some web applications or APIs available online offer user authentication through a trusted identity provider (the most famous examples of this situation are the social network platforms such as Facebook and Google).

An IdP can also be identified as a SSO, as each page under the main page of the login uses the same user authentication provided in the beginning, meaning that a user logging in at the home page, will have his same information available on the pages under home (see figure 3.7).

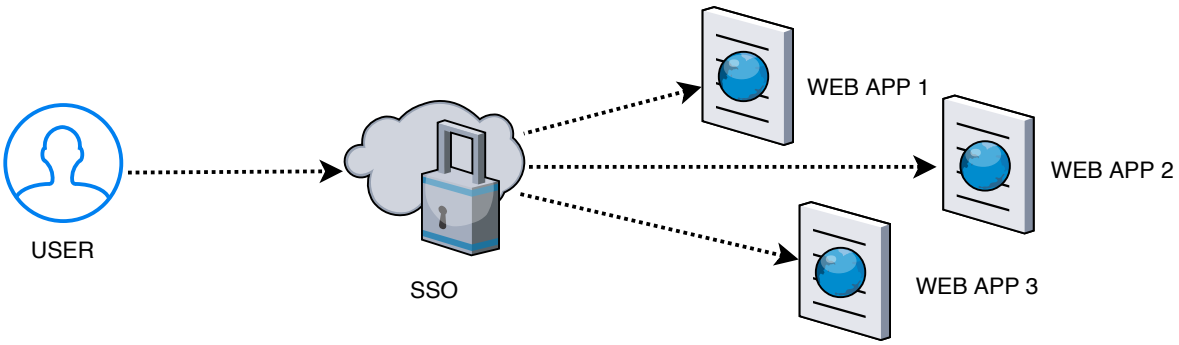

Figure 3.7: SSO working as the access control

3.3.2 Google Sign-In

As users nowadays grow more impatient to use applications and often even disregard any sign-up options, third-parties have invested in "One-Click" Authentication solutions. The Google Sign-In API made available by Google is one of the easiest, safest ways to provide user authentication for users all around. Signing in with a Google account enables users themselves

to quickly sign up to the application without needing to fill out a form that would require the basic set of information that Google already has available.

The user still needs to have - or create - a Google account that is associated with a Google e-mail defined by the user. The information associated with the existent (or created) Google account is used identification purposes, and to differentiate the different data provided for each user when using the different features on the dashboards.

3.3.2.1 How does Google Sign-in work?

Google Sign-in works as a middle-man that primarily recognizes the user that is logging in (see figure 3.8).

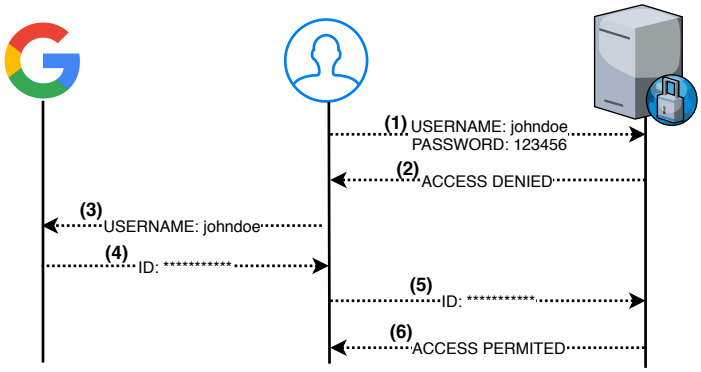

Figure 3.8: Google Sign-In

By offering an authentication method and validating the user as authorized of access, it simplifies the process of signing in for both the user and the system. The Google Sign-In also uses and manages the OAuth 2.0 flow and token lifecycle (the Server generates a token to maintain session) [40].

3.4 Mapping

When it comes to mapping framework platforms, there is a lot of variety available for the user to use: some of them make specific integrations easier (like the Google Maps API), but only a few of those are optimal for mobile use.

So, when it comes to mobile user-experience, one must account for framework size (which overall affects the loading times as well as resources used from native mobile), the simplicity and performance capability, the responsiveness of the platform for the overall system (this would take into account the fact that it's a platform to be used either mobile or desktop), and other factors that, simply put, Leaflet [41] is the one leading on.

3.4.1 Leaflet

What Leaflet offers, is a simple, intuitive platform which is well-documented and is easy to use. It works as a JavaScript (JS) framework, which is optimal for making responsive platforms and works both on mobile and desktop. It is composed of 38kb of JS files (minus the external plugins that can be added), which makes it one of the most lightweight functional

JS frameworks, of its type. When it comes to UI, Leaflet gives the liberty to implement visual information as seen fit, as well as a whole set of interactive features. Since it mainly affects the basic inner workings of map visualization, the rest of the features and add-ons can be implemented by the user.

The developer can handpick the map tiles themselves, which means that different layouts of maps can be used (in this case, OSM was used- section 3.5), providing different visualizations for different cases in need. The customization part does not only affect the actual map tile: Leaflet is capable of producing any marker design possible: it is possible to use a variety of custom icons, either created by leaflet, or by the user. Leaflet also organizes its framework efficiently, using objects such as:

- **Map**: defines a map object given a Document Object Model (DOM) id of a element.
- **Marker**: instantiates a marker object given a set of coordinates.
- **Polyline/Polygon/Rectangle/Circle/CircleMarker**: Each of these instantiate an object given an array of geographical points.
- **Layer Group**: Groups several layers and handles them as one. Each layer can be an object that is overlaying the map.
- **Popup**: Set pop-ups that appear on the map.
- **Control**: A base class for implementing map controls. All controls extends from this class.

There's a lot of different ways maps with the Leaflet framework can be used, and the variety of existent plugins makes it a very versatile tool. On top of Leaflet, there's also the possibility of adding specific functionalities built on top of it. One of those is the one used in this case, the Leaflet Routing Machine.

The Leaflet Routing Machine is an easily extensible way to add routing to a normal Leaflet map. With just a few lines of code, it is possible to build a system that can calculate distances and routes from point A to point B (exemplified in figure 3.9).

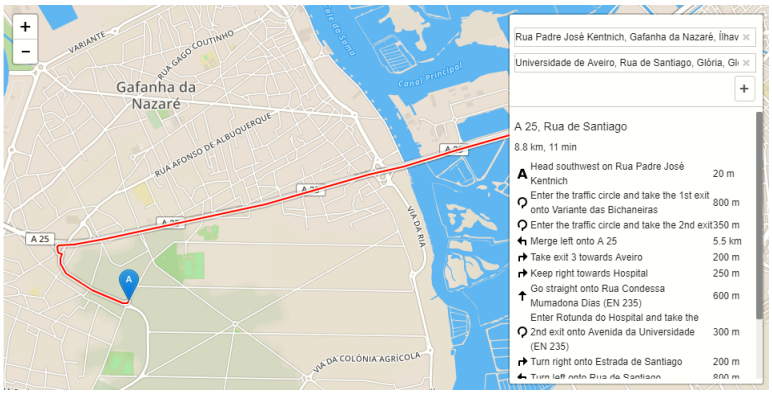

Figure 3.9: Leaflet Routing Machine in action

It is highly customizable and built with multiple features such as:

- Able to edit, remove, or add waypoints through input boxes.

- Accurate, extensive description of the movement to-be-done on the route.
- Support for other several routing engines (OSRM, Mapbox directions API, etc.).
- Open Source.

3.5 OpenStreetMap

OSM is the map that was chosen to represent the visualization of mapping information. There are alternative possibilities to chose from when it comes to Map tiles. OSM though, is a free, open-sourced, collaborative project of data and information gathering. It has a very detailed representation of locations since it mostly gathers its information from ordinary users that want to collaborate.

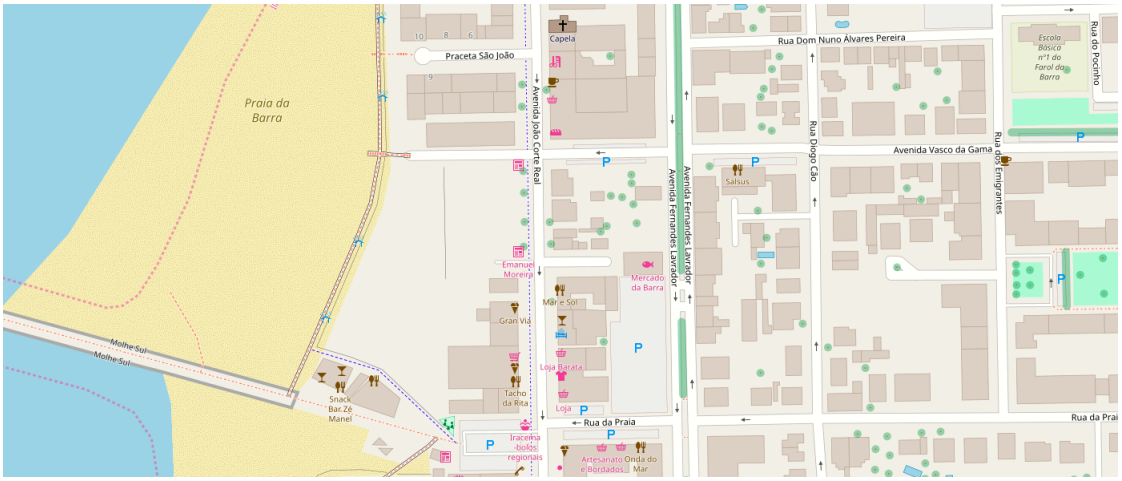

Figure 3.10: Example of OSM Map

In figure 3.10, it is possible to see the level of details as it goes up to parking spaces, public bathrooms, commerce, etc.

This project began as a concept coming from its creator, Steve Coast, in 2004 (the initial focus being the United Kingdom). It is editable (with a specific browser editor[42]) and contains contributions that go up to the governments' jurisdiction.

Among its functionalities, OSM is also capable of doing vehicular routing. This routing is done externally and applied on top of OSM (this includes a small but proper list of resources to use this such as OSRM, Mapbox APIs, etc.).

When applied to mapping frameworks (such as MapBox or leaflet.js), OSM is also able to incorporate whatever features the frameworks themselves possess. This includes their routing capabilities, marker positioning, polygon drawing, and whatever visual representation they use.

OSM annually gathers people from the OSM community in an event called **State of the Map** (a play on words for State of the Art). It has been consistently happening since 2007 and spread all over the world [43].

CHAPTER 4

Implementation

After setting the requirements for the system and taking into account all the details discussed for each page, the implementation of the application took place.

Whereas most of the features presented on the Requirements chapter were implemented, some changes might have been made as well as some adaptations of the proposed implementations to the real thing. After all the technologies needed were taken into account, a system was built on top of a microframework named "Flask", which is set up in Python 2.7.14 (but can be built on with Python 3).

4.1 System architecture

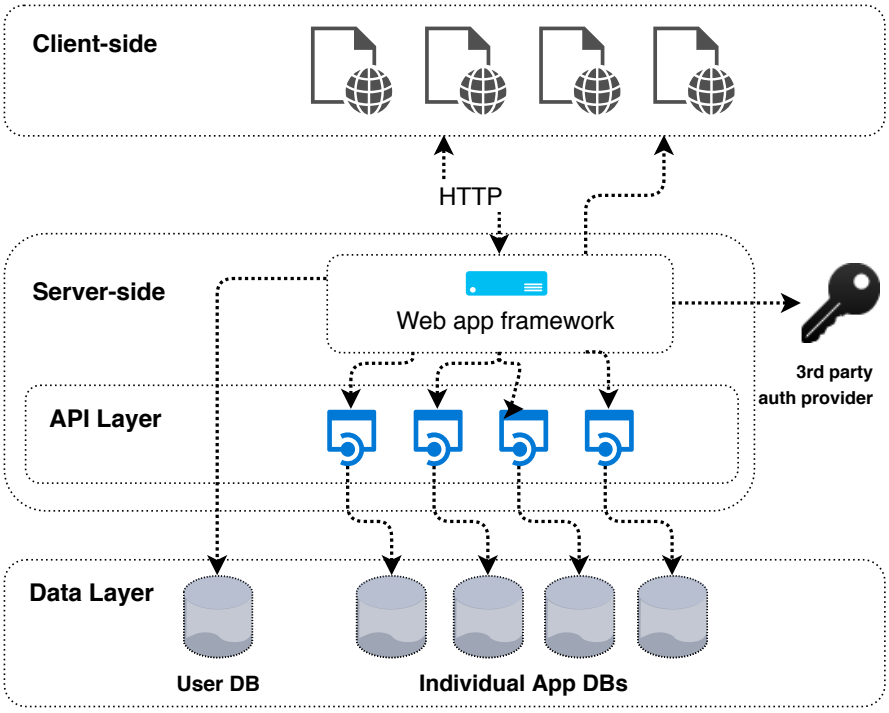

Figure 4.1: System Architecture

The system architecture is composed of three layers, each of these centered around the Server-side, whose *Web app framework* works as the passageway for most of the data going from the API layer and data layer to the client side.

Most of the communications are performed via Hypertext Transfer Protocol (HTTP) methods, while the rendering of the pages is done by tools within the *Web app framework*. The user's authentication is performed via a *third party authentication provider*.

4.2 Server-side

To implement the Server-side aspect of the application, a way to correctly render the HTML Client-side was needed, as well as a definer of end-points to seamlessly capture all information and send it to Client-side.

Python Flask was a tool that, aside from being simple and capable of doing everything that was necessary within the spectrum of this application, is a low-weight library used primarily for web development. It is easy to set-up and easy to start developing, with a well-documented structure [44] and an active community of users that can develop together.

4.2.1 Organization of the Server-side folder

It is important to also keep an "organized desk" of your work- for this, all specific subjects and matters gathered on the Server-side were organized on the standard way for web development.

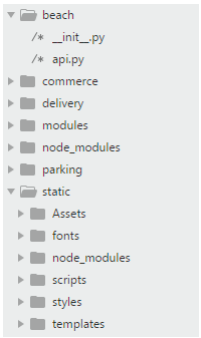

Figure 4.2: File Organization of the system

On figure 4.2, the file organization consists on the separation of the different application folders with their APIs specified, these have access to their data and connections to databases as well as the SCoT platform. The node_modules is the folder containing the different libraries for Server-side exploitation installed via NPM. "Static" folder contains everything regarding the Client-side application. This includes both scripts, styles, templates, fonts, and whichever resources needed.

4.2.2 End-point definition

The main file for Server-side was then created, to define end-points for APIs, Web Pages, handling user requests, etc. It is also here that the Google Sign-in API is instantiated and makes its verifications for the user log-in based on the Google Sign-in.

CLIENT_ID and GOOGLE_LOGIN_CLIENT_SECRET were set according to the project defined targeting this system in specific. The GOOGLE_LOGIN_CLIENT_SECRET was then set as the app.secret_key, which, when set, cryptographic components can use this to sign cookies. *Socket.io* was also set and connected to the *app* variable defined at the beginning of the file, as well as the URI to the user database to manage the logins made using Google Sign-in.

The end-point defining the root of the application performs a verification of the Google user logged-in using the object "Session", which allows the developer to store information specific to a user, enabling the requests to be performed with the logged user's information. Since this is implemented on top of cookies, these are also signed cryptographically; hence the users could look at the contents of the cookie but not alter it unless they are aware of the secret key defined earlier (as GOOGLE_LOGIN_CLIENT_SECRET). If the 'google_token' is saved in session, it means that session_info is available and therefore a user has signed on the system and thus his information can continue to go from request to request untill he logs out. The verification is done: if true passing the information to the Client-side together with the rendering of the index, if not, variable will be transported as false (see code 3).

```
@app.route('/')
def root():
    if 'google_token' in session:
        session_info = session['google_token']
        user = session_info
        return render_template("index.html", username = user)
    return render_template('index.html', username = False)
```

Código 3: Root end-point.

The remaining end-points define the URL locations of each page, as well as the API end-points which fetch the individual data that will consequently be fetched via JavaScript using XMLHttpRequest, which is one of the most used tools to communicate and interact with Servers. It is heavily used with Ajax Programming and can receive/send not only XML data (as the name would imply) but any data, and while mostly used with the HTTP protocol for communication, it can also support file and File Transfer Protocol (FTP) protocols. On Server-side most of the end-points that end in api, are targeted with GET methods, meaning they only serve the purpose of providing information, instead of receiving it.

4.2.3 Google Sign-in

One of the few exceptions to this case is the login end-point: This particular case can only receive information, which makes it a POST target for information coming from the Client-side. When the user logs in using the Google Sign-in, a token is generated with relevant authentication data and sent to that end-point. Finally, using the *google.auth.transport*'s Requests and *google.oauth2*'s id_token [1] the verification of the account is made with the POST message containing the token which can be transformed and used for the authentication. This

[1] *google.auth.transport* and *google.oauth2* are both part of the Google Auth Library for Python

ID token is compromised of a series of some important values that only upon authentication can be decoded:

- **"iss":** The iss value in the ID token is always equal to accounts.google.com or https://accounts.google.com.
- **"sub":** After making sure the token is valid, it is possible to access the sub value a.k.a. the user's Google Account ID.
- **"aud":** The value of *aud* is equal to one of this app's client IDs. It is important to check this value to prevent ID tokens being issued by remote apps being used to extract or access data on our app's Server.
- **"exp":** The expiration time of the token.
- **"email", "email_verified", "name", "picture", etc:** It's also possible to later on access to profile information gathered from the user's Google Account.

All of this data composes an essential part of the verification of the user as it is sent to the backend Server. The *iss* value is one of the first values to be verified, to check if the value contained in the token ID sent to the backend is indeed a Google account, and therefore issued by Google itself (see code 4[2]).

This operation has been standardized by the Google Sign-in API, for any other operations to conclude this process are unnecessary and mostly irrelevant. After verifying that the user exists, the generic data regarding e-mail, name, and picture are registered in the user database in case it doesn't exist already. The user itself is saved on the Session object to save the information and the user that is currently logged in and to still be able to make requests with the user's information stored.

```
idinfo = id_token.verify_oauth2_token(token, requests.Request(), CLIENT_ID)
if idinfo['iss'] not in ['accounts.google.com', 'https://accounts.google.com']:
    raise ValueError('Wrong issuer.')
```

Código 4: Google issuer backend verification

After the verification is done, if approved in all fields, it is then checked if the user exists in the application's database. As the application itself contains values that need to be associated to a user ID- which in this case, the user ID is automatically Google's sub value- during the login verification a verification is perfomed to check if the user exists and, if not, to register them.

4.2.4 API connection

After discussing the main file for Server-side end-point definition, we will break down the connection to the data available for each page. On figure 4.2 there is a folder for each page implemented on the system. The point of this organization is to divide the gathering of information dedicated for each page smoothly. Since each folder contains the API file, one of these, for reference, will be discussed, as most of all these similarly normalize data.

[2]The "token" variable under verify_oauth2_token() is the ID token sent from the Client side to the Server through the POST method.

For testing and further example of the application's functionality as a whole, and since real-time data wasn't available, dummy-data[3] was created to test the system's limits as well as capacities. For reference purposes then, the Parking API will be the one referenced.

The Parking API presents the connection to a dummy-data database, present with data that includes the following structure:

The database has one table element named *parking_space* with three useful elements:

- **lat**: is the latitude of the parking space's location.
- **lon**: is the longitude of the parking space's location.
- **avl**: is the boolean value of whether the parking space is available at the time or not.

The Parking API mostly takes care of normalizing and treating data so it can be sent to the Client-side in a readable format. It connects to the database, fetches the data from it, and performs all the necessary operations.

This database is highly mutable (on a *field*, practical mindset) on the *avl* value, as it will constantly change according to the current state of parking on location. For testing purposes, a small MQTT protocol based API was created to simulate mutable data that changes from a set of time- this, in practice, would send data as the values changed instead from a periodic time. Since MQTT works exceptionally well with small capacity sensors, it was the most suitable protocol to operate this testing phase on.

Thus, while changing the *avl* data from a period set of time at random (using *update_parkings*), information was sent to an end-point that was then called from the Client-side to showcase the data.

Each of these APIs has a job to secure a connection with their database and consequently treat data so it can be readable for the Client-side developer who's then going to expose said data to the typical user.

These functions provided by the APIs are imported on the main Server-side file and exposed to their own end-points under "/api/x[4]" (so */api/weather* or */api/commerces*, and the remaining pages).

The main Server-side file also presents communication with the User database, being that it is the center-point of connection to the remaining pages, as the log-in is meant to be used, single, and common to all other pages under root.

4.3 Client-side implementation

Here we'll divide all the implementations and talk about each page individually.

[3] Dummy data is information that does not contain any actual real data, but instead shows how data would be presented in a hypothetical manner- this is particularly useful for testing environments and operational purposes.

[4] x being the different pages available

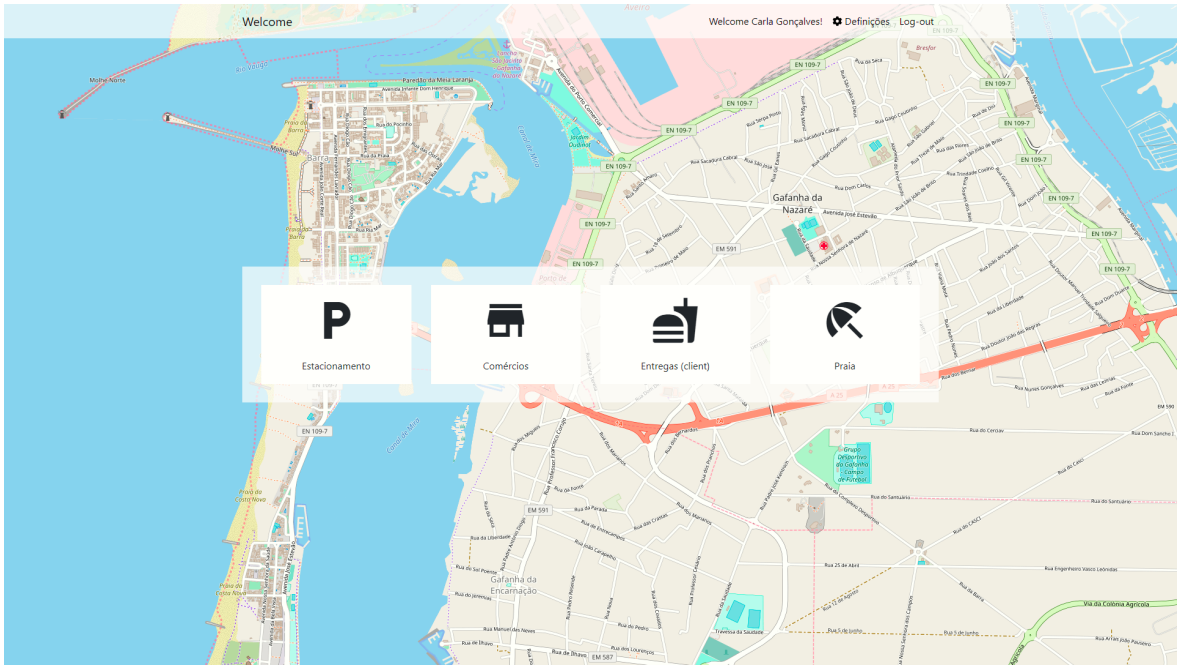

Figure 4.3: Current live page of the Index page

Starting with the primary, index page, a central hub to display all the accessible pages was built. It contains the *log in* button, and the four distinct accesses to the remaining built pages (see figure 4.3).

4.3.1 Parking page

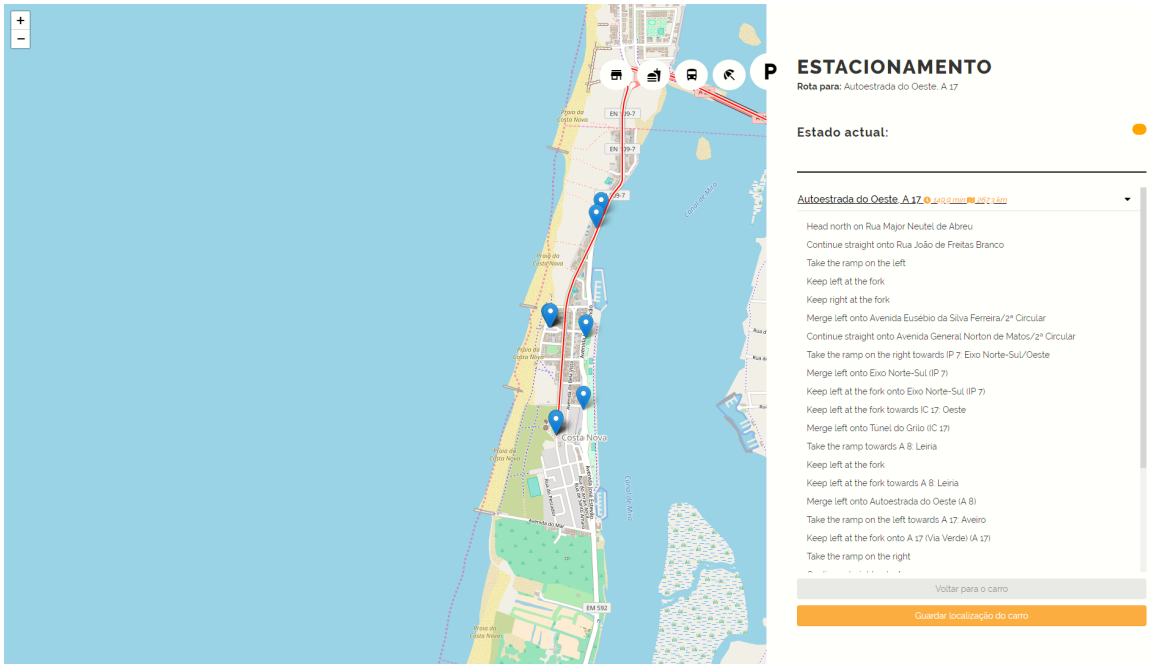

Figure 4.4: Current live page of the Parking app

For this implementation, besides having to receive real-time data on the clock, we would have to look into presenting said data in a readable, quick way. For this, *socket.io* was considered. *Socket.io* is a tool that is bi-directional for communication between Server and Client. This is excellent for this type of application since it works as a tunnel of sorts and a bridge to communicate data on the fly. Since MQTT itself works as a publisher/subscriber but isn't the most obvious way to apply communication directly to the Client-side, *socket.io* comes here in the form of communicating to the Client-side while "subscribing" to the MQTT publisher. Upon receiving the information, *socket.io* instantly transmits the information and performs operations on a message received.

With this in mind, all the parkings received through *socket.io* are instantly mapped on the map view display of the parking page. Only the parking spaces with "avl" equal to 1 are mapped, as they are the available spaces and the only ones worth showcasing to the user. This information is also *Broadcasted*[5] as all users have to receive the same information, since no exceptions are made.

Another feature of the Parking Page is the *Current state* of the parking spaces. This is a treated data set that is formatted on the Parking API from the Server-side. The total number of parking spaces divides the number of available parking spaces and later multiplied by 100 to get a percentage format of the average number of available spaces. After that, the data is compared to the following schema (Percentage of available Parking spaces is (...)):

- **Lower or equal to a third**: Shallow capacity, so it returns the color-code "FireBrick", for "red" state.
- **Bigger than a third and Lower or equal to two thirds**: Medium availability of spaces, returns "Orange" color-code.
- **Bigger than two thirds**: High availability of free parking spaces, returns "LimeGreen".

The reason a color-code is the value sent by the function that goes by *get_spacing_capacity* is so that the Client-side will automatically assign the style to the element that is showcasing the availability of the parking spaces (this feature is implemented on the element showcased on figure 4.5).

Figure 4.5: Current state of parking availability

Another feature is the *Save car location* button, which, after the user eventually parks their car, it enables them to click on this button and consequently save the last location they were at upon clicking. This button, on-click, sends the last location it was found at as well as the user data needed to assign that very location to the user it belongs to. This process is done to avoid confusions regarding whom the "last" location belongs to. After the button is

[5] this is set on the *socket.io* transmission, which means that the message transmitted is set to *Broadcast = True*

clicked and the POST action is done, the button is instantly disabled, and the *Go back to car* button makes itself available, as a location was saved and can now be reached (figure 4.6).

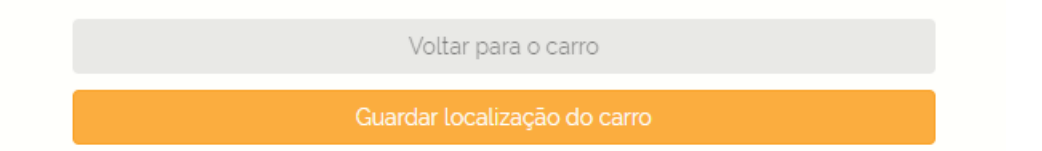

Figure 4.6: *Go back to car* and *Save car location* buttons on their initial, unchanged state

After analyzing all cured data coming from Server-side and being exposed to the Client-side, it is possible to look at the features implemented exclusively on Client-side.

These are all mostly Map-related operations as it is our central participant in this page in particular. For starters, the state diagram of this page is particularly helpful, so it is possible to figure out all the operations it is capable of (see figure 4.7).

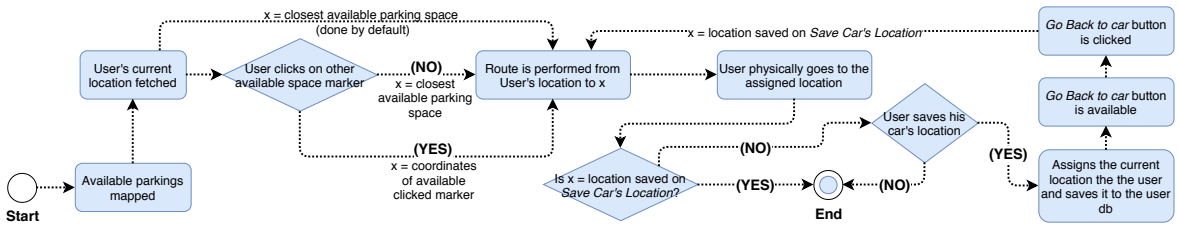

Figure 4.7: Parking State Diagram

On the figure 4.5 above, it is possible to infer all the capabilities of this application when it comes to Client-side functionalities. This state diagram assumes per definition that all communications between Server and Client are already made, and doesn't specifically identify the data traversing since it has already performed that action.

Thus, the features implemented on Client-side, what they can do, and how they have been implemented can be discussed.

The User's current location is fetched using the *Locate* Geolocation options which are defined on the Geolocation methods provided by Leaflet.

Geolocation[6] contains two simple methods that consist of *locate* and *stopLocate*. These in itself do a standard job in fetching the current location of the device, using for this the *Geolocation API*, and firing a *locationfound* event when successful, or a *locationerror* on failure. The *locate* options, however, are the discerning factors that offers the most features for this type of application: The location can be set to *watch*, which means that it is constantly looked up when it changes coordinates on the user's part. The *watch* option itself is set using the W3C *watchPosition* method, which consequently also offers an *enableHighAccuracy* method.

After the location is successfully fetched, the system automatically routes the user to the nearest location available- this in part generated some controversy among the evaluators, since it decides for the user. It was implemented, however, thinking of the initial state of the page

[6] *Geolocation* and the methods within it are part of the Map object specified in *Leaflet's* API

as well to showcase the abilities of the applications (to manage and generate routes, that is). Nevertheless, after the location of the user is caught, the system selects among all the parking spaces that have been marked which one is the closest. The code organization of this principle is to call a function which receives all the locations, sorts them, and ultimately selects the *closest* one to the user as the *default*. After this, the user is meant to intuitively figure out that he can click on each marker and obtain the same results: the routing from the user's location to the assigned point (all this process is demonstrated on the interaction diagram on figure 4.7).

To do the routing, *Leaflet Routing Machine* is used.

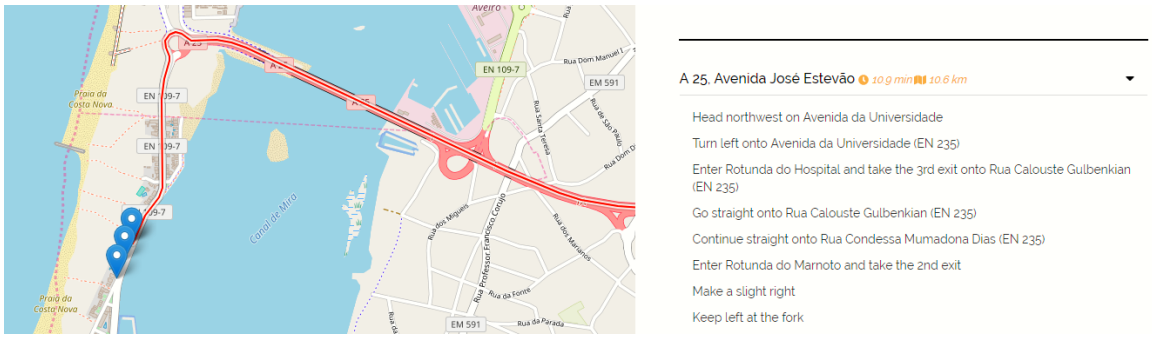

Figure 4.8: LRM being used on the system itself

On figure 4.8 it's possible to observe the live approach of the implementation; On performing the routing operation, a red line that represents the route from point A to point B (presumably A = user's location and B = Parking space marker) is drawn on the map, and the information that details the *movements* the user has to make from their location to their destiny is provided by *Leaflet Routing Machine*, fetched, and put on display on an accordion-like collapsible item on the primary sidebar of the page (right side description). The information regarding *the street their destination is located at* is given, as well as the *time* it will take them (by car, by default), and the *distance* they're at (these details can be seen next to the name of the destination street on figure 4.8, in yellow).

Finally, after the user eventually parks, they have a choice to *Save the car's location* (or in the practical implemented sense: save the last location they are at and assume they use this tool for their actual purpose), or do nothing and go about their way.

4.3.2 Beach page

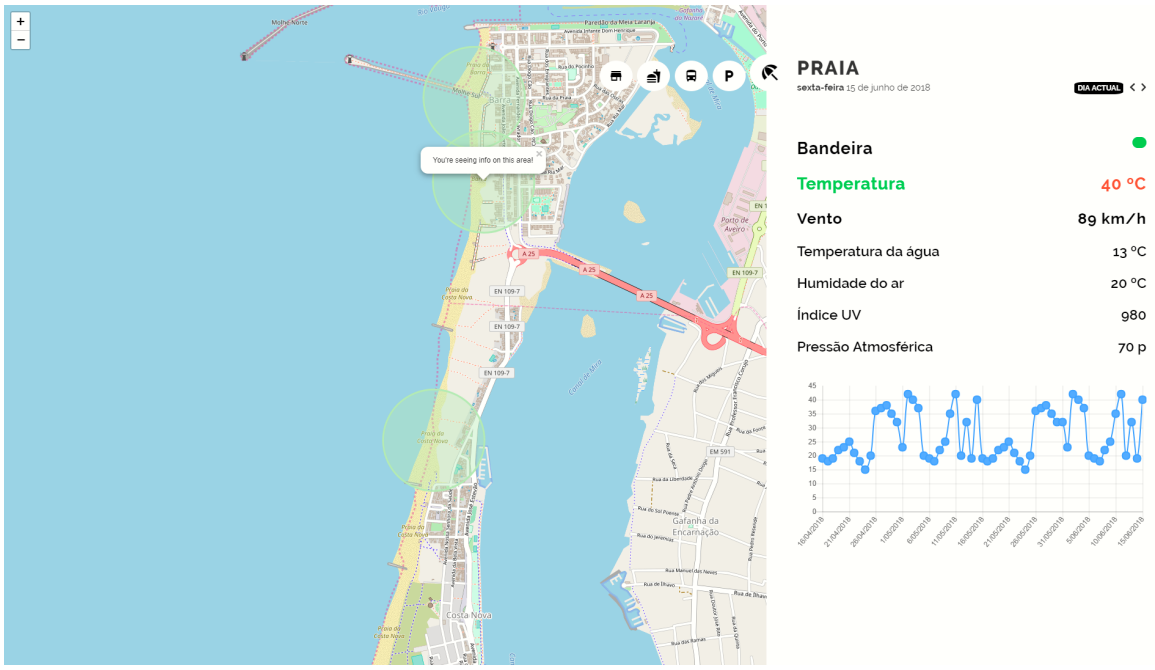

Figure 4.9: Current live page of the Beach app

The Beach page focuses primarly on information representation. It was distinguished from the very beginning to expose all the possible information, but find a way not to overwhelm the user the minute he entered the page. It's useful to have so much information, especially regarding weather data- not only for typical users who'd make use of this theme in a more diminutive manner (a.k.a. only look at the temperature and flag information, or, at most, look at the wind information- which actually for some evaluators was an important factor) but also to people that need to gather this particular type of cured data for investigational purposes. It is also interesting to be able to look at all this data in a range-based type of view, not only to look at it but also make conclusions about the variation given at different times of the year.

The data retrieved from the Beach API has different results according to the different places it is going to be showcased.

So to take a look at the use-case for the data (again, this time, dummy-data was used for testing purposes but always written with an easily exchangeable pattern so the real sensors can seamlessly substitute the format tested), such was divided in the following patterns:

- Assuming that there's going to be multiple sensors with a limited range each one- this can either be a good or bad thing:
  - **Bad**: The range will be limited and won't have as much coverage as a normal weather station (assumingly- there isn't at the moment any standardized range for weather stations in general, as it greatly depends on the location, topography, and

landcover[7]).

– **Good**: It presents very accurate data on the described location, as well as a fixed radius of temperature and sensorial factors on-field. The more limited the coverage of the sensor, the less expensive it is expected to be, and thus it is possible to gather more sensors to spread on the beach area and present them on-map.

Assuming then that multiple sensors with a limited range will be available with different (but likely not very disparate from each other) data, it's possible to find multiple solutions to this, as the data could be averaged out and merely present the average of all the sensors spread throughout the area, or more interestingly, it's possible to expose all those sensors and their locations so the users will be able to pick their specific spots to go to, with the temperature and data they find most adequate.

The challenge here was to expose this data, so it was obvious how to look at the different sensor's data and check where they were located. For this, and going for the same kind of theme the Parking had (as will all other pages, since they're in a SPA environment and have a uniformized look), the Beach page will have a map section that will compromise most of the page, seeing as it's the chief navigator for the user.

Sensors are exposed with a Circle colored path available as a marker from *Leaflet*, set with their assigned radius and having *popup* information[8] telling the user that he is *"Seeing information on this area!"*. This implementation can be seen on figure 4.10.

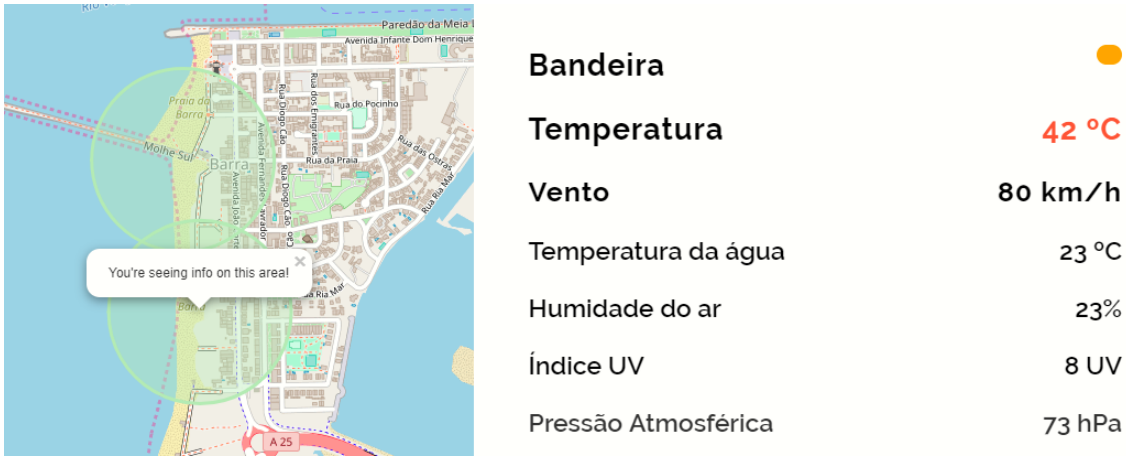

Figure 4.10: Beach Sensor representation & data

Since the Flag state at a given the moment is something that can only be known if the information is user-provided, it is assumed that some outside physical entity provides the information. Any generic weather sensor can quickly retrieve all the other information.

Each different sensor made available for the application contains the same type of information with the only variant being the data values and the locations. With this type of implementation, we're able to see the same **type** of information in **different locations**.

[7]Landcover is the very physical material at the surface of the earth. The term is mostly used on satellite-dependent use-cases

[8]Basically their legend

The weather API divides the data into two different sets:

- **All weather data registered for every day, or divided in periodic times:** This is for graphical purposes, so it is possible to observe how the information has changed from time to time. This information is automatically exposed if the user hovers over one of the legends in the sidebar (i.e. figure 4.10's *Temperatura*, *Vento*, or any other factor). The title of the information the user sees (on figure 4.11) is highlighted and changes color to inform the user of the target of information they are currently looking at.

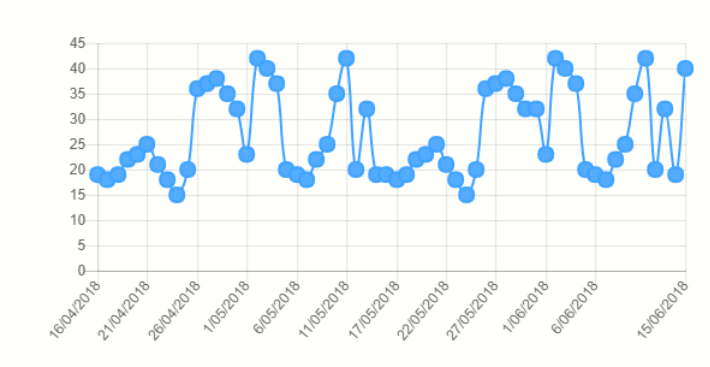

Figure 4.11: Temperature variation from a period of time

All this data, when fetched, needs a value for variable "sensor", where the Client-side asks information of a specific sensor[9]. All the sensor information is also fetched on the weather API.

- **Specific day weather data:** This particular case is for setting the data the user is looking at on the sidebar on first look. The weather API accepts two variables for this, *sensor* and *timestamp*. *Timestamp* is sent from the Client-side and set by the user's desire to view the previous day's data or the next day's data. The default information is always the current day's. The selection of the day of interest can be made on this section of the page, under the title:

Figure 4.12: Specific date data retrieval

Since this page is only informative, the user will only be able to take his own conclusions and make use of the data available, although the primary goal of usability is to offer the users a chance to choose which area they'd like to go to based on the information from different data in **different sensors**, as well as **plan their trips** to the beach **days ahead** and **check the weather forecast**.

[9]The sensor with *it explicitly* one, or rather the first registered sensor is the default sensor set for data visualization the minute a user enters the page. *Sensor* value changes from default on click from the user's feedback

4.3.3 Delivery

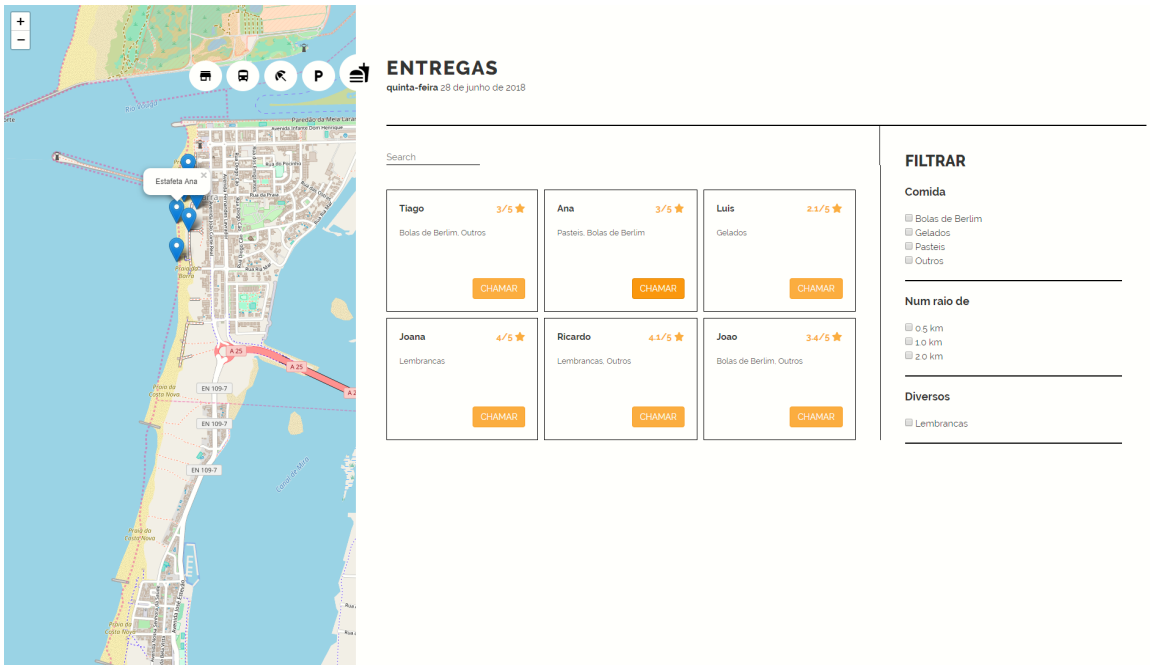

Figure 4.13: Current live page of the Delivery app

The Delivery page contains information that is mostly user-driven. What this means is that all the information that is going to be showcased can in some form or another have interaction with the user experiencing the platform. For this purpose, we will begin to analyze the features available.

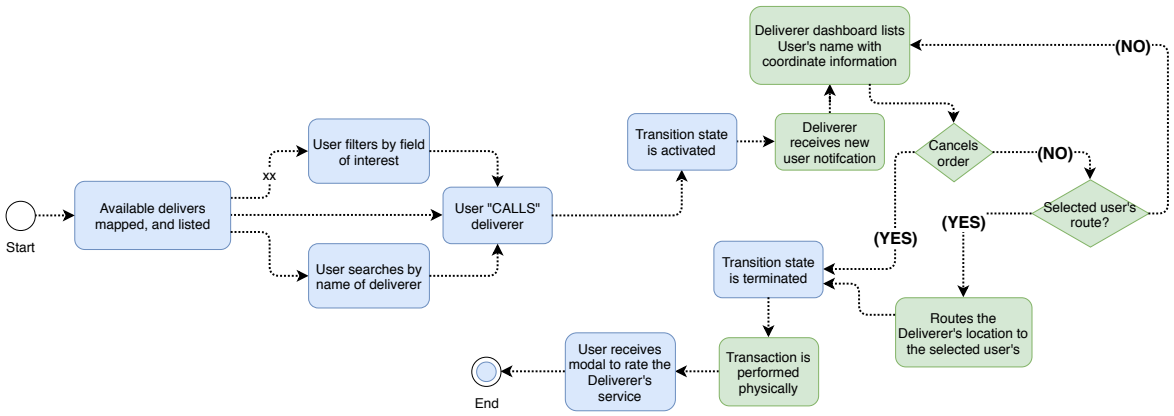

Figure 4.14: Delivery State Diagram

The Delivery state diagram in figure 4.14 represents both the universal user perspective as well as the response in actions[10] from the Deliverer dashboard. The concept of this page has been explained before: it is a simple call-and-response system that contains different points of view due to the present *Stakeholders*.

[10]When specifically mentioning "in actions" of the *user* perspective, it means that it doesn't cover the Deliverer's dashboard, which in itself contains features and operations of their own.

As the available deliverer's markers are mapped on the map section of the page, they're also listed on the sidebar to showcase the different products each one has to offer- this is defined by each deliverer individually, by text separated by commas (this is the current implementation, but it can be subject to change) (figure 4.15). Each of these items that the Deliverers sell is a single product. Since the theme of the subject at hand usually has the same type of products for sale, we have divided them into sections.

Figure 4.15: Deliverers marked and listen on the Delivery page

- **Food:** Food is the most regular type of product sold by deliverers whose territory lies on the beach area. Most of these sell regional food, and others sell summer-themed food, beverages, pastries, or anything that they would find fitting. Since these sellers are registered and licensed to sell products around the beach, it is up to their choice what kind of product they want to sell. If a product happens to be more regularly sold, it ends up deserving its tag[11]; otherwise, it is identified with an *Others* tag.
- **Diverse Objects:** This can range from beach/city memorabilia to sun-wear or anything that isn't within the range of consumables.

That being said, after this field-analysis, we have found that the main articles for sale (Food or diverse) are, as follows, *Bolas de Berlim* (a Portuguese regional cake-like sweet), *Ice-cream*, and *Pastries*. Everything else that does not identify in one of these fields goes by *Other*, and it is up to the Seller to identify the consumable he is selling. These identifiers are the catalysts of filters applied on top of the listed Deliverers, together with another field "*In a radius of...*" that goes from values of 500 meters to 2km (although these are subject to change).

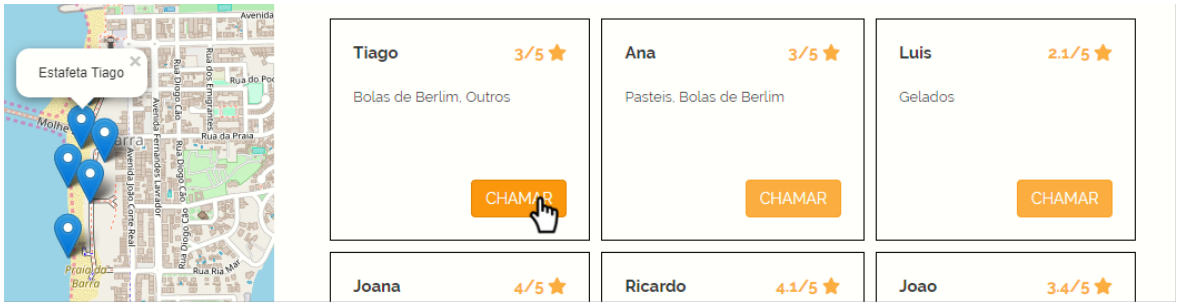

Figure 4.16: Hover action

[11] Or, for implementation purposes, its *filter* checkbox input

As the user hovers through the *CALL* button, the pop-ups of the deliverer they are looking into show up on top their marker, so the user knows where they are situated at the time (see figure 4.16). This also works by hovering on top of the markers.

The Search Bar works by filtering the results on mouseUp, and the ordering dropdown is capable of ordering the list by *Name*, *Popularity*, *Distance*, and *Last Inserted*.

After the user selects a Seller and calls them, the process of delivery takes place and a trigger is activated on the User's dashboard signaling the transaction is in progress (as seen in figure 4.17)

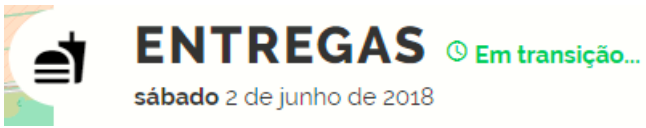

Figure 4.17: Active transition signal

From the other side, the Seller is notified and receives the notification in the form of a new user being listed on his dashboard.

The Seller's dashboard is a relatively simple one- it displays options for the seller to change their type of product being sold, as well as events for the users that have ordered from them. It is composed of a simple Ordering filter that can list the users that have requested them by *Time of request*, *Name*, or *Distance*. *Name* is not exactly helpful for usability purposes, but it is a standardized way of ordering lists that are more often than not applied in cases like these.

Furthermore, after a User requests for a target Seller, their marker signaling their location is placed on the Seller dashboard's map, as well as listed on their sidebar (as per standard) (figure 4.18).

Figure 4.18: Deliverer dashboard: single item of User that has made a request

Figure 4.18 also shows how the User that has requested the service is presented to the Seller: with minimal information and only the one necessary so that the Seller can adequately interact with the person that has ordered from him. Some details may be added posteriorly, but so far the standard is as follows. The only data available is the name of the User, their coordinates, a routing button that instantly maps the route from the Seller's location to the received one from the User, and a *Cancel* button, if the Seller is unable to accomplish the task physically.

Assuming that the User accomplishes the task of ordering and the Seller moves towards him, and towards transaction completion, after exchanging goods and currency, the Seller can end the transaction, which automatically sends a message to the User to rate the service.

Rating service is based on a simple score from 1 to 5 *stars*, and it is calculated according to User satisfaction. After rating, the transaction is fully complete, and service ends. The

goal of using a rating system is to ensure good quality service for the user- having a way to score the service provided for them ultimately imposes pressure for the seller to have proper service standards.

4.3.4 Commerces

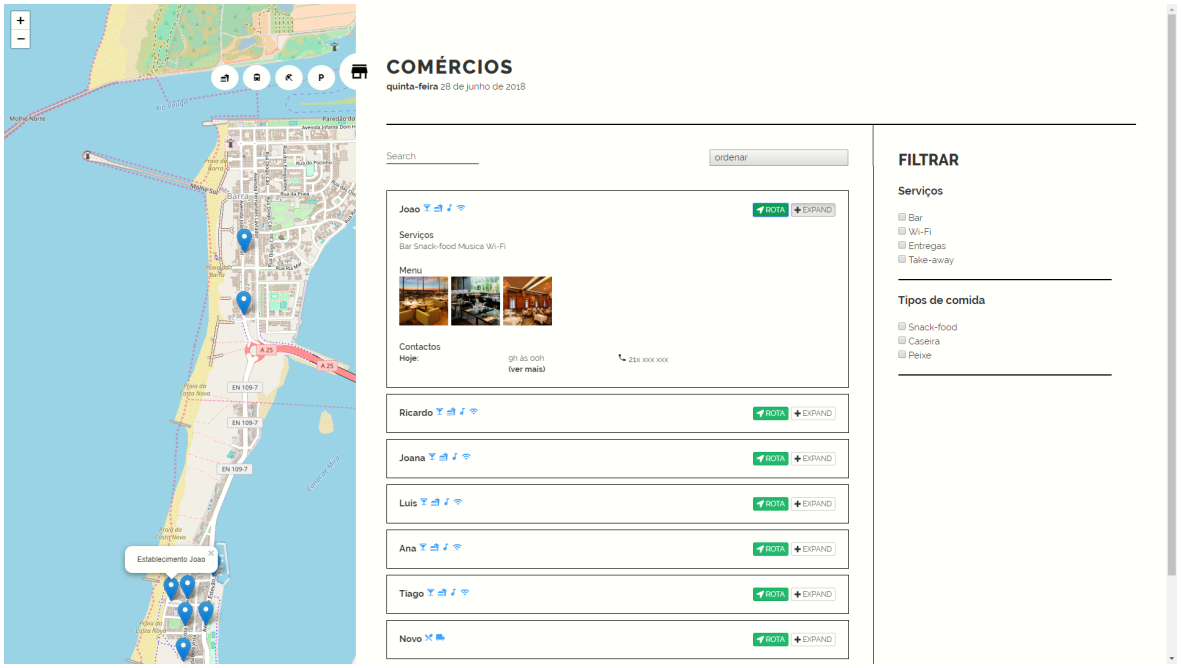

Figure 4.19: Current live page of the Commerce app

The commerces page surfaced as a necessity, and per request of inquiries made at the beginning of this project.

Most of the ideas for application implementation were gathered from people that would be most likely to use this type of thematic system (beach-goers, local people, and others). One of the most requested implementations were surrounded around bar and commerce information surrounding the area. People were most interested in knowing about the price ranges of certain bars or restaurants, opening and closing hours, among other possible information the Commerces would forthcome in providing any and whichever information they wished to expose to the public.

The system itself is following a theme that not only aims to help improve but also *give* what the users wish for. It implicitly raises and generates economy exchange - if a user knows enough information about specific commerce they had doubts about, they would be more willing to visit it, and partake in their services. Whatever commerces that showed interested in having their businesses shown on this application, would be able to do so by providing the data they wish to expose to the people responsible and behind the project (this, as all elements referenced previously with this solution, can also be subject to future alteration - as the system will eventually be able to grow more independently from its creators).

The primary information asked from these businesses is their selling, defining points. The *Name* of the business, *Type of business* (bar, restaurant, and if restaurant, what type of food

they're serving, if they provide music or any other services they might see fit to expose), *precise location* (or coordinates of said place, for implementation purposes), and *pictures* (of the place itself or the menu). All this information is provided by the Business Owners and thus it is of their interest to assimilate what they wish to show to the typical user to put the best focus on their commerce as possible[12].

Each commerce is saved with all this information, with the coordinates being used to map them out on the map section of the application (figure 4.20).

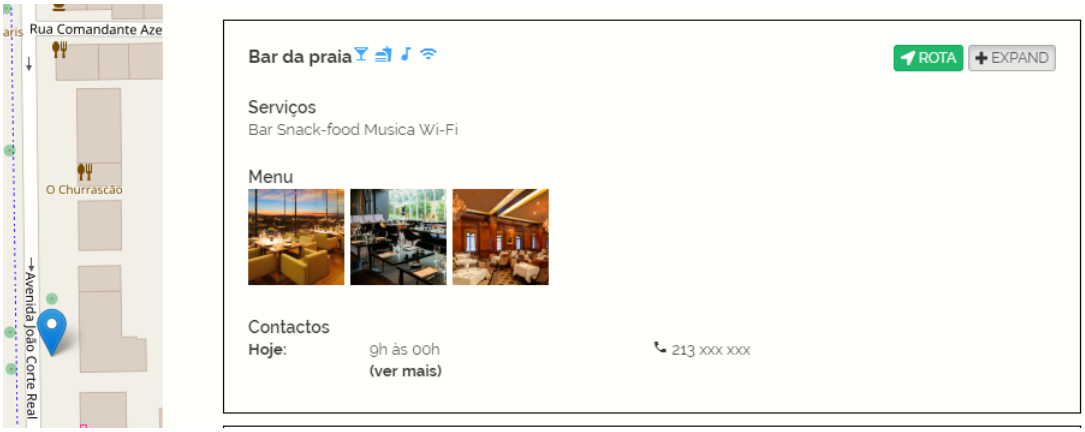

Figure 4.20: Live Commerce Page with the detailed information of a single listed business

The application is also built to enable the user to, on the click of a button (on figure 4.20 this button is called *Rota*), have easy access to the location of the selected business, and have a route traced for him from the user's location to the business. The list of items is also listed in an accordion-type of format, with the option to expand the information on each commerce. To facilitate usability, the users are also able to intuitively figure out what type of services each business provides by the Iconography used and displayed even when the item is not collapsed.

The filtering system works much like the Delivering page does, in the sense that most of the common services are listed individually otherwise it is grouped on *Others*. The filters are also divided into three sections:

- **Services:** Which symbolize the type of additional services the business offers. This can range from *Wi-fi* providing, *Bar*, *Delivery*, *Take-away*, etc.
- **Food type:** Food type is for those that mostly identify as restaurants, but can also be applied to bars that serve *tapas*, or more known as *Snack-food*. Food type can range from a lot of different food specialties: *Regional food*, *Fish*, *Grilled/BBQ*, etc. All of these have much potential to grow and expand regarding variation.
- **In a radius of...:** This was also a filter option available on the Deliverer page. It detects the user's location and checks which business is closest or within the range selected by the user.

[12] Reasonably, it's safe to assume that the more a *services* a business owner showcases, the more chances it will have of being *found*

The images available on the *Menu* information section are expandable and work as a modal image gallery- or a *Lightbox*. On click, the pictures expand and can easily be looked at without reloading the page.

4.4 Mobile Implementation

As the project itself identifies as a mobility solution, a mobile adaptation with the same features presented previously was developed.

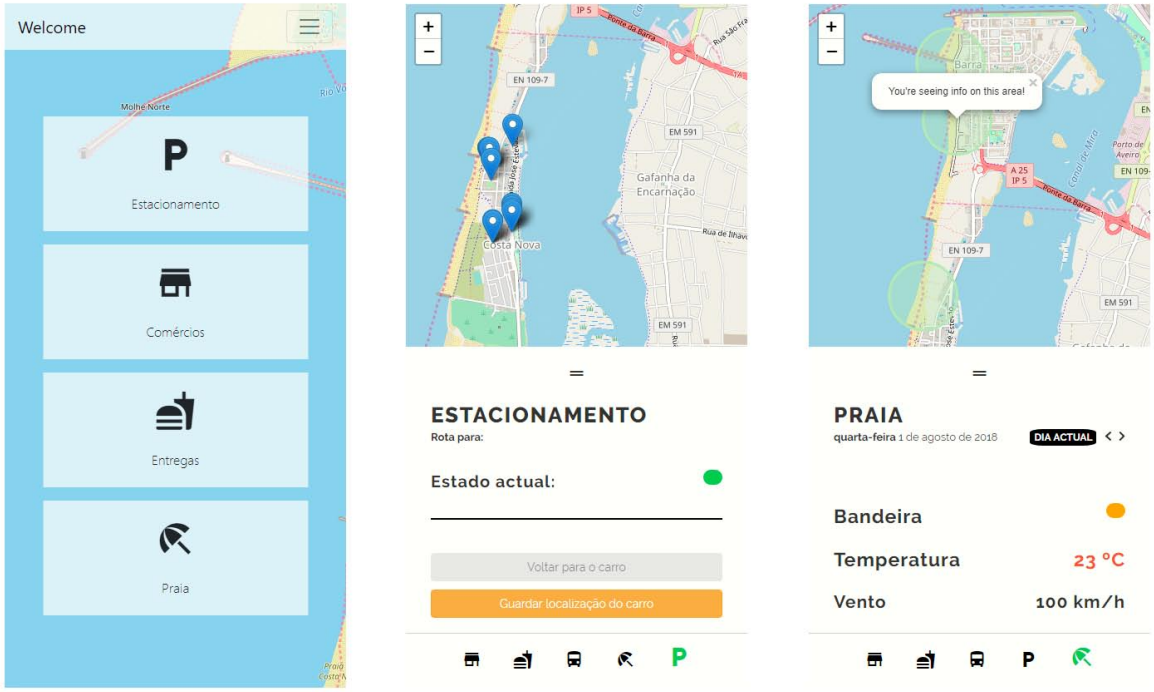

Figure 4.21: Mobile Adaptation of the previous pages

The UI represented was heavily influenced by other applications such as *Google Maps*, which offer similar disposition of elements as the ones present here.

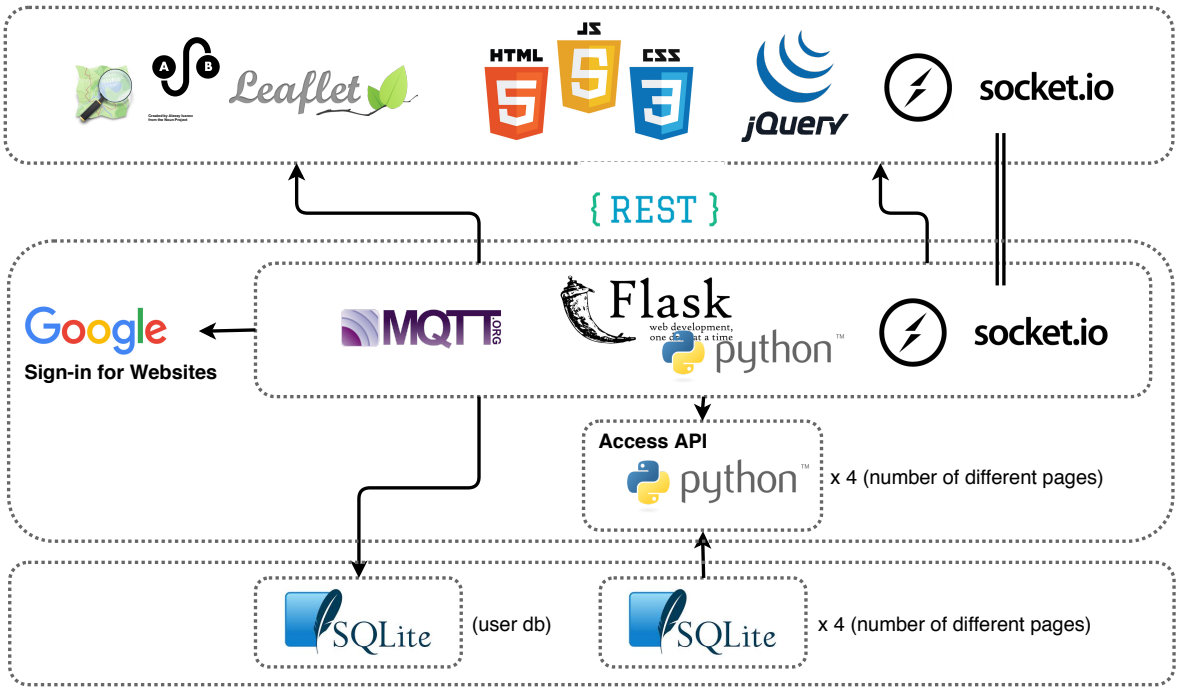

Figure 4.22: System architecture and technologies used

The system architecture is composed as above-structured on figure 4.22. Each of the technologies will come into context as the functionalities of the system are described. For now, a succinct description of the system:

The upper layer is composed of the Client-side of the system, with technologies such as:

- **HTML/CSS/JS**
- **jQuery:** a cross-platform JS library designed to ultimately simplify the actions and scripting of JS on client-side.
- **socket.io:** used to establish bi-directional, straight-forward communication between Client and Server.
- **OSM, LRM, and Leaflet:** used to create the map view, calculate and draw routes, and attach markers and other user-friendly features.

The middle layer represents the Server-side. The Server-side contains the communication with Client-side, log-in authentication (using ***Google** Sign-in for Websites*), end-point definition, and access APIs for each page with their specific logic for treating data originating from each of their databases. The access APIs are the primary connections to the databases each of them is the owner of; using four separate databases for each page to maintain full independence from page to page.

- **MQTT:** used to establish a publisher/subscriber within the Server-side.
- **Flask/python:** the web framework as the main structure for development (i.e. end-point definition, rendering of web pages, etc).

- **socket.io:** used to establish a bi-directional connection between Server-side and Client-side.

The third and last layer defines the *data layer*, where user information and information from each page is stored into separate SQLite databases.

CHAPTER 5

Usability Testing

Given the context of this application, an inquiry was provided in regards to the usability of each available application aimed at the participation of different types of users:

- Technology-focused people (or power users) are people that use this type of technology regularly[1], as well as know the standard buttons that are used for the same type of actions regularly (in this spectrum, search bars are usually present in a consistent style, and people are quick to find these elements as they are used to it, same goes for filter buttons, navigation elements, etc). This set of people are meant to find specific elements of a page more easily than non-regualr users of technology systems.
- Non-users of technolgy identify as people that are the opposite of the previously described, hence don't identify as users of pages that are technology-driven and have UI elements that are unfamiliar to the user. It is expected that these users have more difficulty in using this type of applications.

After evaluating the type of users that are to be testing the usability of this application, there is a need to identify the optimal way of performing these tests. The question comes to mind: What is being evaluated? The usability of this application relies heavily on the intuition of the user to find some aspects of the interface presented before them. A narrative is set to help identify what elements are more accessible to the user and what type of decisions they commit to these tests.

Among a set of different methods for evaluation available for the usability testing, the primary form of testing used in this case was through Cognitive Walkthrough and the Heuristic method.

5.1 Cognitive Walkthrough

Cognitive Walkthrough is identified as a usability inspection method[2] primarily used to identify interface-wise issues and interactions within the interface, focusing this way how easy

[1] Also known as *power users*

[2] Usability inspection is the name for the set of methods where an evaluator inspects a user-dedicated interface and classifies the different elements presented

(or vice-versa) the reachability to individual elements and/or tasks is. It is task-specific, as in, given a particular task or a narrative to the evaluators, they evaluate within a specific metric how feasible the assignment was. This method takes into account the users' capacity to reach certain elements or complete tasks without the need of an actual manual for utilization- this is especially important for user-focused applications or web services whose target is to, simply put, simplify a whole process that was, non-ideally, complicated.

Cognitive Walkthrough also prides itself in generating accurate, fast results with real-life experimentation and low-cost test generation as it spends very few resources and only needs a small set of users to obtain accurate results relating to the usability of the system.

To perform these tests, a set of variables were taken into account:

- How easy it was for the user to follow the narrative given and reach their goals.
- The total time the user spent performing the task from beginning to end without external aid.
- The assumed time from the user (as in, how much time the user thinks they spent to accomplish their goals).
  - The comparison between time spent in reality and time spent presumably is a strong indication of how easy or intuitive a task is. Assuming that it exists three case scenarios for this data:
    - **Time presumed < time in reality:** this means that the user has found the navigation and task to be easier than it was if they think they spent less time than they did performing it.
    - **Time presumed > time in reality:** if an evaluator thinks he spent more time browsing and performing the task than he did, it indicates confusion regarding initial navigation.
    - **Time presumed = time in reality:** although hard to pinpoint, it happens, and the results taken from this scenario are mostly inconclusive.
- How many clicks took them to reach their accomplished goal.
- How many features were used to reach the goal (i.e., if they made use of features such as "ordering", "filtering", "search", etc.).
- The understanding of the presented visual aid elements and how helpful they were to the goal as a whole.
- Understanding if the user feels at any time at a loss of what to do to accomplish the goal given by the script.
- Constant dialogue and monologue from the user to understand what steps are most obvious and what vice-versa.

The Cognitive Walkthrough was then divided into different sections for the different pages, beginning from the Parking page, Weather, Delivery, and Commerces in this order.

5.1.1 Cognitive Walkthrough Script for Evaluators

The tests were made with the evaluators present along the owner of the project, to accurately transcript the dialogue and the train of thought users go from the start of the evaluation process.

The goal of this script was to take the evaluator into a specific mindset and test the system as it went, offering a use-case scenario that could happen to a simple everyday user from start to finish on their day. The script was built with the purpose of being immersive enough to let the user sink into the idea that he was the one performing all the sets of actions, justifying why they were performing the tasks, as well as enabling the user to find the solutions to the problems exposed on their own accord. The script was as follows:

The goal is to test this application that's vastly dedicated to beach-related services, four of them are currently available for your personal usage, although the usability is to be tested in a web-desktop environment, assume that it is also usable from your mobile and the same features will be available in a more portable device than your computer (if it makes it easier, you can assume this usability is dedicated for a tablet-type of device). From the various applications at your disposal, perform the set of tasks that are given to you.

You, the user, want to go to the beach, your vehicle of choice in this situation is your car. Although you travel by car, you know that there's a grand possibility that the parking surrounding the beach area is going to be mostly occupied, making it nearly impossible to leave it somewhere close by. Nevertheless, you know that more often than not you can find a parking space (even if in a semi-secluded (assumingly) but reachable area). En-route, your location is detected and always watched over, so the system is aware of your current location at all times. The goal here is to make as most use of the application as possible, so, given this and with the aid of this application, how were you to find an available parking space to park your car? Please narrate the steps you take into accessing your parking space of choice.

Found it? Awesome! You're all set, but alas, unfortunately, you had to leave the car in a seemingly unknown place to you- you're not used to this location even if it is a secure and normal one. What about when you return? What will do after parking your car?

If your answer was "I'll memorize the place and pray that I'll find it. My car even has a remote lock, and it blinks when I press it so if I'm near it, I'll certainly find it!", maybe think a little more practically. Unfortunately in this scenario, your car is a bit older than you want it to be, and it doesn't really have a remote lock which you can detect at a small distance, the area is easily forgettable (or maybe even ***you*** *are a bit forgetful), and no, you have no external aid in memorizing this place. Make use of the resources you can find! What is the first thing that comes to mind to solve this problem?*

Figured it out? Great. If you haven't, good luck finding your car, I'm sure you'll still find it.

After parking your car, you are headed to the beach. Usually, people are quite picky as to the place they're going to settle down. Are you one of those people? What are the main factors of your interest? Could it be temperature? Usually, location features such as having much available sand-area or leveled terrain with minimal dunes are one of the choices for casual beach-goers. Unfortunately, this type of factor is given to people that already know the area, but locations like these are usually preferred due to their temperature- maybe areas with a lot of dunes have a higher temperature, or even less wind, given that, what's your criteria of choice? You have a lot to chose from. Or maybe you don't even care at all. Nevertheless, you

are going to want to go to a higher temperature area- this time of the year is perfect to get a good tan. So you check the temperature of the location.

Decided where you're going to lay your towel? Is there any area with a better temperature? Maybe less wind even? Figure all those factors before you settle down.

I assume you're all set up on the spot you've chosen, but now you're hungry! What do you do? You're all alone, and you've forgotten to bring any type of snacks with you (you certainly seem very forgetful lately). What do you do? Maybe you can wait around if any of the delivery people around the beach pass by you. Maybe even buy a Bola de Berlim or an ice-cream (those are especially in season). Waiting is so tiresome though, don't you agree? Do you see anything that could make this process any faster? Check your applications and take into account if there's anything available to you.

(Yes, I mean the Delivery page)

Found it? What are you in the mood for? Maybe try to go for some ice-cream. What're your criteria for choosing the deliverer to come to you? There are so many people that deliver ice-cream distributed across the beach, pick your favorite!

After you've chosen your deliverer, he comes to you, delivers your food and wishes you a good day. You eat it all up and are happy and satisfied, another well-served customer! So you rest for a little bit and hang around the beach for some time.

Time passes by, and without even noticing, it is already 1 pm. Hunger strikes again. Maybe it is time you pack your things. This is a job a mere Bola de Berlim cannot accomplish, you're hungry for more, evidently.

So you pack your things, open your application and see if there's anything more this fantastic system can offer you! The first thing that comes to your attention is the Commerce page that is available to use! You find it intriguing. Go to it.

Alas! There's a whole set of places you can eat! What are you in the mood for? Home-cooked meals? Maybe some fish, maybe even some snack-foo, or "tapas", as they're known. You're not really in the mood to eat at the restaurant though, you're way too tired actually to eat properly- instead, you choose to take your food home and eat on your couch, watching some TV and immediately taking a nap (what a pleasant way to end the day, no?). Pick your elected restaurant, and find out where it is. Is there a way to reach it faster? Sure you know it's close to... that other place... but how do you get there? You're vaguely aware of your current location but haven't got an actual idea of where you should go to reach it. Try to reach your selected restaurant more easily.

I assume you've reached your restaurant and got your food. Excellent. You're all set for a delicious lunch and a fabulous rest after a sunny, beautiful day of beach-going.

You walk out of the restaurant and realize you have yet another problem. Where's your car?

You have two options here:

1. ***You saved your car's location**: Excellent! You know what comes next, so find how you can reach your car.*

2. ***You didn't save your car's location and since you are indeed a very forgetful person, you're destined to search for it forever and end up eating in the street:*** *Well. At least you got a good tan.*

That's it! Thank you for testing this platform, we hope this has been at least enjoyable for you and that you'll be able to use this platform in the future as your guidance for any beach-related activities in Praia da Barra and Praia da Costa Nova.

5.2 Heuristic Evaluation

Another type of evaluation is the Heuristic method. One of the heuristic laws available are the 10 Laws of Usability by Jakob Nielsen, which are one of the most highly-regarded sets of heuristics even used today (and the catalysts of developing a heuristic-based evaluation). Among those, IBM and Apple also present a good example of industrial design reference.

Apple, for example, partakes in the discipline of developing applications that *serve human beings* or *positively affects the lives of the people that use the developed apps*. Rather than focusing on building a *beautiful* or *simple* application, responding to the emotional and practical needs of the person it is being designed for, is the main goal [45].

IBM is involved in three main practices that compromise of:

- **IBM Design Research:** Which in itself is composed of principles that mainly unite the *user* and the *business needs*, encourage teams to follow the user, measure success, and exercise endless curiosity. *IBM Design Research* is the step where services and products that empower client success are delivered [46].
- **IBM Design Language:** A set of vocabulary for doing *great design*. Also composed of a front-end development vocabulary, which is meant to compliment the design vocabulary, to deliver great user experience aimed for *your* user [47].
- **IBM Design Thinking:** A framework for modern enterprises that re-envisions *design thinking*, and focuses on the speed and scale of the outcome, providing this way better solutions, in less time [48].

A heuristic evaluation is also considered a Usability Inspection Method, as it is focused mostly on evaluating the interface part of the project (focus on UI). The point of the heuristic evaluation is to take the used set of heuristics and evaluate if the interface of the system complies with the rules described. These are considered one of the most taught methods of usability evaluation across the world in education.

Following the core and original set of heuristics, it was decided to follow on Nielsen's set of heuristics, as they are arguably the most complete and thorough set that has been put to the test for over 20 years of usage [31] .

1. **Visibility of system status:** The system must keep the users informed of what's happening.
2. **Match between system and the real world:** The system must have the same language as the user with familiar concepts.

3. **User control and freedom:** Users must be free to choose their operations, as well as always have a way to return to their initial state.
4. **Consistency and standards:** System must leave no room for ambiguation regarding words, situations or actions.
5. **Error prevention:** Rather than having good error messages, it is best to avoid them altogether.
6. **Recognition rather than recall:** Objects, actions and options should be visible and best identifiable by consistent looks to facilitate the user's task and not overwork their memory.
7. **Flexibility and efficiency of Use:** The system should be fast and smooth to use.
8. **Aesthetic and minimalist design:** No more information than the one necessary within the whole page.
9. **Help users recognize, diagnose, and recover from errors:** Error messages should have the actual messages; no codes or icons describing the issue.
10. **Help and Documentation**: If ultimately necessary, the user should be able to find a way to contact additional help either from the source or application documentation.

The heuristic is a useful method of evaluation because it offers the evaluator a chance to grade the system as a whole- rather than performing task-based actions, the user is put in a perspective of judging the system for what they should and shouldn't have, and of checking whether or not the platform they're testing is consensual with the laws proposed by Jakob Nielsen.

5.3 Usability test results

5.3.1 Usability test - Background of participants

A selection of people at random performed both the Cognitive Walkthrough and the Heuristic evaluation, among 19 different people whose background is composed of:

- **Average age** of 21,5, with a range between 18 and 27 years old.
- 100% of the evaluator's **occupation** is **student**.
- 32% of the evaluators are **female**, with the reamining percentage being **male**.

The tests were divided into two sections, where they read the Script given first, performed their tasks and were timed by the project owner at each break[3] of the script. The tasks were usually performed in a straight-forward manner, with minimal breaks besides the ones related to the task (i.e., if the person was confused, it would take more time).

Some of the tasks, as the pages had a minimalistic design and straight-forward information, were seldom performed together (by the evaluator's error). The fact that some applications proved to be so intuitive to the user and sometimes even caught the interest to perform multiple operations when in fact, they were supposed to be done separately, had some users *skip* some of the tasks in theory, when in reality they were just performed out of place from the cognitive script.

[3]Break here can be considered the break performed from the Script to look at the application so they could perform the task suggested by the ulterior

5.3.2 Cognitive Walkthrough - Setting

The results of the Cognitve Walkthrough method can be divided by page, and the reviews were not so different from everyone that performed the tasks, which makes the final report of the experience more accessible to organize.

According to the script, actions are not exactly specified, as it leaves freedom to the user to chose when and how to act to complete the actions according to the script telling them how they would act out a normal day. There's **nine** total breaks:

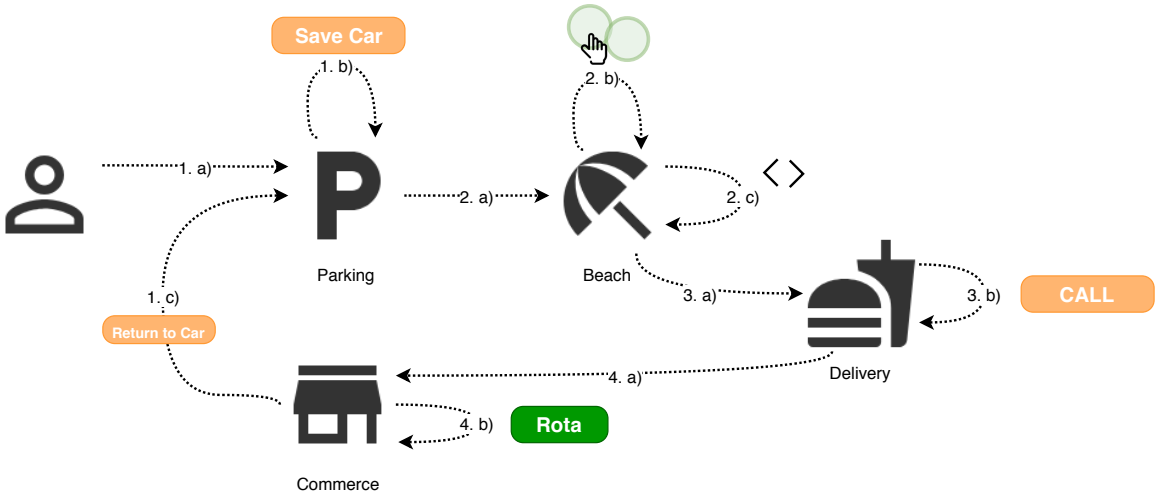

Figure 5.1: Usability Test Walkthrough State

1. **Parking:** 3 tasks.
   a) Go to Parking Page and consider all information that can be found and retrieved for your usage.
   b) Save the car's location.
   c) Retrieve the car's location taking into account that step 2 was performed.
2. **Beach:** 2 tasks (and a half) [4].
   a) Find the Beach page button and go to it. Look at the information presented and make conclusions.
   b) Click on the circles available on the map side of the page, these circles should represent different sensors in different locations, with different temperatures and information in general, the goal was to chose the one with the highest temperature, or whatever factor the user chooses to have as the decider of where they'd do their beach activities.
   c) [5] Let the user find the buttons they would have to click to find the date-picking (i.e., Go a day forward or backward to check if the weather has been pleasant, or is going to be pleasant tomorrow).
3. **Delivery:** 2 tasks.

[4] An action that wasn't on the script is also experimented by the evaluators
[5] The "half" action

a) Go to the delivery page and gather all information presented to you, the number of delivery people available, the filters, etc.

b) Chose a deliverer. You can either go from the filters and select the one you want or go directly by scrolling through the list. After you pick one, "call" them by clicking on the "CHAMAR" button and wait for the order to process. Look at the new information that appears and make conclusions.

4. **Commerce:** 2 tasks.

a) Go to the commerce page and perform the same actions as the ones presented on task 1 from the Delivery page. Look at all the information available as well as the iconography exposed on the page.

b) Pick a place where you'd like to eat according to the guidelines given to you by the Script. You can either use the filters, iconography present or simply by expanding the different listed locations and looking at the information in them. Click the "Rota" button to get the route from your location to the one selected.

Since the pages are simple enough for regular usage, there's only so much a user can do to experiment all the applications to their full extent and to make the situation believable. The primary goal was to show as much information as possible, but still keep it simple enough so the user would not get lost on it.

5.3.3 Cognitive Walkthrough - Results

As described previously, since some of the steps were done together, a brief description regarding the usage of each user is going to detail the average for each step.

- For **1.Parking a)**, there was a divide of about 26% of users that thought the Markers were not legible enough, i.e., they should be more specific as to what they represent: instead of a simple blue marker, some popup with information regarding that car could be used. 74% of users thought it was legible and instantly understood that they represented free available spaces to park.

  Regarding the "State" information, it was mostly not legible and regularly misunderstood, as 50% of users found the current state to represent different values (such as weather condition at the moment, whether the sensors were online/offline, and others) incorrectly. Thus, some suggestions to make it more clear in terms of *wording* were made by the evaluators themselves.

  Most of the users thought they took significant less time to execute the action than the actual time it took them, meaning that it is possible to conclude that this task was more straightforward than they would expect.

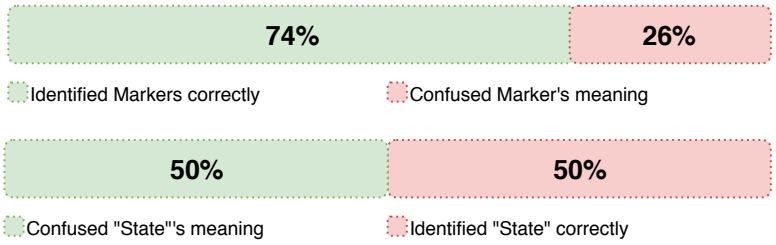

Figure 5.2: 1.Parking a) results

- Step **1.Parking b)**, to save the car's location, 30% of users were not able to easily locate the button, either because they were confused by the language, or didn't find it at first. Nevertheless, 90% of users, including most of those that were confused by the step itself, managed to solve the issue quickly, usually lasting about 5s to do this task, and all of them guessing the time it took them correctly.

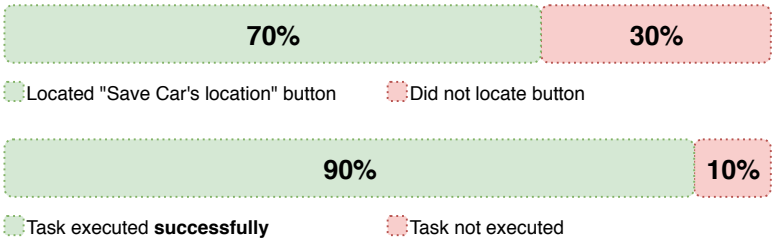

Figure 5.3: 1.Parking b) results

- Step **2.Beach a)** had some particular hardships regarding the usage of the map for interaction- something that hadn't been approached before on the script until this moment. Nevertheless, 90% of users found the beach page accessible with numerous interesting information. The main challenge was to make use of the icons directing them to different pages on the page they were at the moment (Parking). 80% of users had no difficulty using this icon, while others returned to the previous page and navigated to the Beach page through Home.

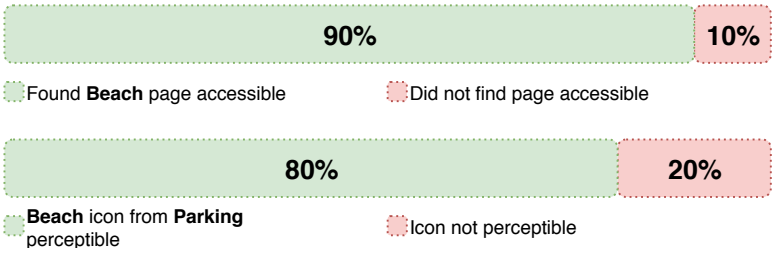

Figure 5.4: 1.Beach a) results

An average of 80% of users thought they spent less time executing an action than they had, which we can conclude by saying the application looked more straightforward to use than it was perceived.

- For step **2.Beach b)** around 20% of users executed this action together with the previous task. For the ones that didn't, they concluded that it had a good intuitive and

perceptible utilization. Some users suggested to color-code the areas described in the beach area to differentiate them and lead the users to want to click them.

Figure 5.5: 1.Beach b) results

Most users took more time to execute this task than they predicted (60%).

- The results of step **2.Beach c)** were very straight-forward with 100% of the people correctly accomplishing this goal.

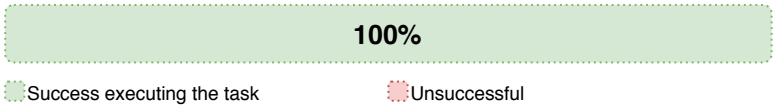

Figure 5.6: 1.Beach c) results

- Step **3.Delivery a)**, which had the users selecting the Delivery page icon, had about 80% of the users easily reach it, with a perceptible icon and easy straight-forward navigation. There was a case where the evaluator suggested the names to be visible on the page (together with the icons), as well as one other user who mistakenly clicked the "Commerce" icon.

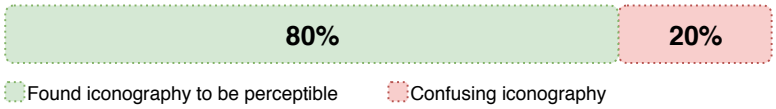

Figure 5.7: 1.Delivery a) results

Regarding time, 80% of users thought it took them less time than it took, and the remaining percentage thought it took them more time than it did (one of these evaluators goes on par with the one that mistakenly clicked the "Commerce" icon).

Iconography was a solid point of evaluation for this page- one of the evaluators, a person with dyslexia, heavily relied on these to navigate and evaluated these as very perceptive and helpful for the overall navigation of the page. This person, in particular, found the navigation throughout the page very intuitive, with little to no hardships.

- On the **3.Delivery b)**, where the task was to make an order based on criteria, 30% of users performed this task together with the previous one, as they reached the page. To the ones that didn't, the 70%, 50% made use of the filter tab and correctly called for their "Deliverer" to come to them. All 100% of them performed this task correctly.

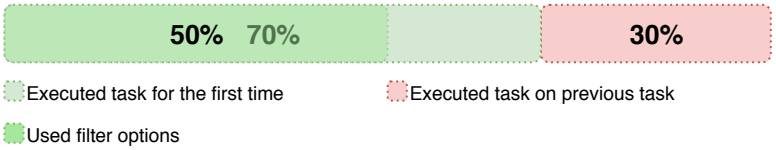

Figure 5.8: 1.Delivery b) results

Based on time, 70% of users thought they took less time than they did, 10% guessed their exact time, and the remaining 20% perceived it as taking longer than they did.

- **4.Commerce a)** Had an easily reachable icon to navigate to. Often this icon had trouble being confused with the "Delivery" icon due to their similarities in theme, even though they had different functionalities. Other than the regular occasion and evaluator confused those two icons, most of the information and navigation ran smoothly.

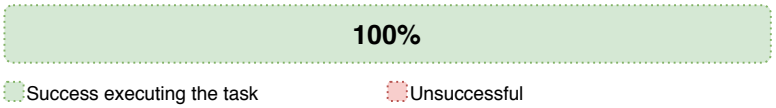

Figure 5.9: 1.Commerce a) results

The time it took for the users to perform this task was mixed between the evaluators guessing the exact time, performing the task in less time than expected and vice-versa. The results, regarding time, are thus inconclusive.

- The 8th task presented to the users **4. Commerce b)**, had the goal to find the restaurant with the criteria present in the script, and to correctly press the "Rota" button to route the users from their location to the restaurants'.

  For this task, most users used the filtering option and used the button correctly. This task was executed correctly 100% of the time, while the filtering function was only taken advantage of from about 40% of users.

Figure 5.10: 1.Commerce b) results

With this said, 80% of users assumed that the task took them less time than it did, with the remaining 20% correctly guessing the time they spent on the action.

- Finally, the last task of the script had them return to the very first page they operated on. This was to test the easy recognition and remembrance of prior situations. On this final step, users had to go back to the parking page and click on the *Voltar para o Carro* button so they could get the initial coordinates that were saved by them on step **1.Parking b)**. Thus, in step **1.Parking c)**, 96% of users performed this action in a quick, swift manner, quickly remembering what they were meant to do from the previous recognition of the Parking page. The remaining people did not perform the task in **1.Parking b)** in the first place.

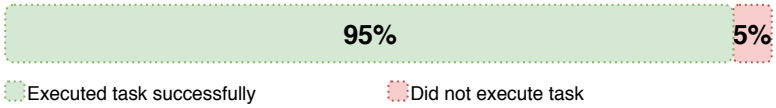

Figure 5.11: 1.Parking c) results

All of the people that executed this task operated in either less time than they thought they had from their guess or correctly guessed the time it took. This discrepancy in time can have an origin on the evaluator assuming that, because it took them some time to think of where they would perform this operation, browsing to the first page they used took more time than it did. The difference in time assumed by the actual time was never more than 7 seconds.

5.3.4 Heuristic Method results

In this section, all the users that evaluated previously on the Cognitive Walkthrough followed up by executing the Heuristic Method, using the Ten Laws of Usability by Jakob Nielsen [31].

Nineteen evaluators performed their score of the heuristic laws and rated them from 1-10 (10 being the highest), compared to the system they were evaluating. On average, the results were as follow:

1. **Visibility of system status:** 8,3
2. **Match between system and the real world:** 9,3
3. **User control and freedom:** 8,7
4. **Consistency and standards:** 8,2
5. **Error prevention:** 8,4
6. **Recognition rather than recall:** 8,2
7. **Flexibility and efficiency of use:** 9
8. **Aesthetic and minimalist design:** 9,6
9. **Help users recognize, diagnose, and recover from errors:** 8,4 (only 57% of evaluators gave judgement on this heuristic)
10. **Help and documentation:** Non-Applicable (N/A).

For proper visualization of the number of times a specific classification is scored from all users on each heuristic, we have created a heat-map of sorts to demonstrate the dominant classification for each rating. It should be self-explanatory, but the bigger circles are the dominant classifications, whereas the smaller ones are the rare scores.

The results on the literal scale were as follows:

5.3.4.1 *Visibility of system status*

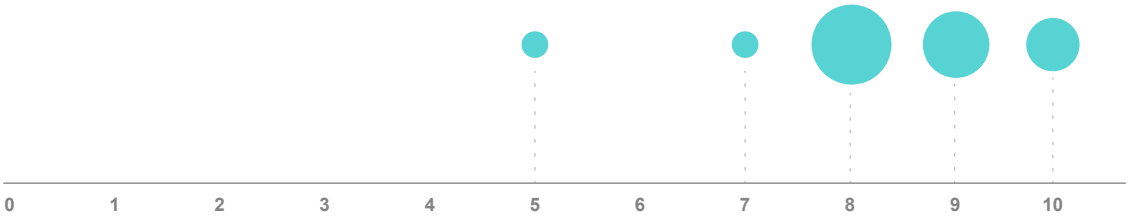

Figure 5.12: Number of scores for *Visibiity of system status*

The results on this heuristic were a bit disparate; the predominant classifications is *8*, which while still high, it is still not as ideal as one would expect. The results remained between 5 to 10, which means that it balanced out on a positive scale from the evaluators, with little low numbers, and prevalence from 8 to 10 (about 79% of classifications sit on this range).

From this analysis, it is safe to assume that most users felt comfortable on the page they were navigating at for most of the time, as they were assured of the current status, and aware of the functionalities each offered.

5.3.4.2 *Match between system and the real world*

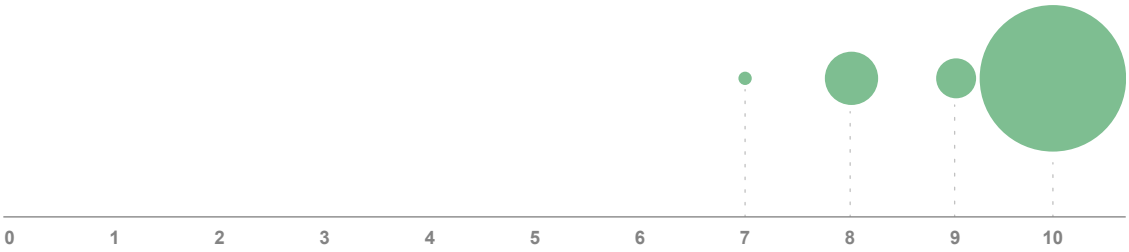

Figure 5.13: Number of scores for *Match between system and the real world*

Compared to the previous heuristic evaluation, the results coming from this were far more apparent - 58% of users gave it a perfect score, while 100% were higher than a good average of the *7* classification. The results for this heuristic were, in conclusion, very positive.

5.3.4.3 *User control and freedom*

Figure 5.14: Number of scores for *User control and freedom*

Almost half of the evaluators classified this heuristic with a perfect score. Besides one classification of *4*, all of the remaining testers presented a score of *7* to *10*.

Users found that control and freedom was one of the features that were most cleverly implemented, and only lowered their score due to some missing elements to navigate through the system.

5.3.4.4 *Consistency and standards*

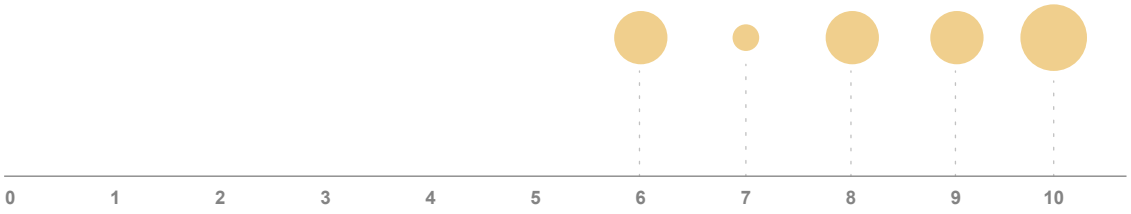

Figure 5.15: Number of scores for *Consistency and standards*

Consistency and standards mostly envolved the Iconography presented on the interface. Most users found that the icons were clear and legible with a positive classification ranging from *6* to *10*.

The SPA type of implementation also helped to keep a consistent look of the whole interface, which ultimately affected the final score given by the users.

5.3.4.5 Error Prevention

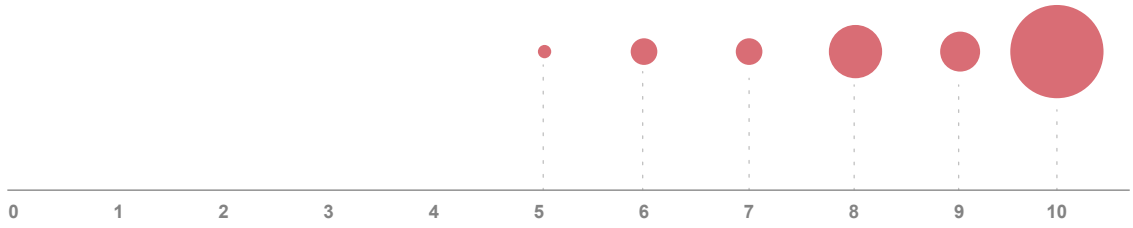

Figure 5.16: Number of scores for *Error prevention*

This heuristic also had a majority sitting on numbers from *8* to *10*, with about 74% of evaluators giving a score within this range. All but one user had classifications above the average *5*, with the singular evaluator specifying that this heuristic was not so clearly represented in the interface before them.

5.3.4.6 Recognition rather than recall

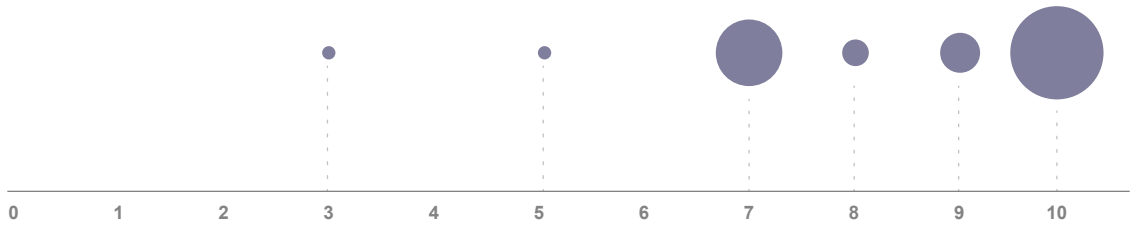

Figure 5.17: Number of scores for *Recogniton rather than recall*

Recognition rather than recall was also heavily represented by the Iconography of the interface- Recognizing certain elements and the familiarity of the icons regarding the subject of the page were vital in this evaluation.

Approximately 74% of evaluators deemed the interface easy to follow through *recognition*, while 11% of the results were average to low (the lowest being a *3*) on pair with heavy criticism for the icons and representation chosen for the system.

5.3.4.7 *Flexibility and efficiency of use*

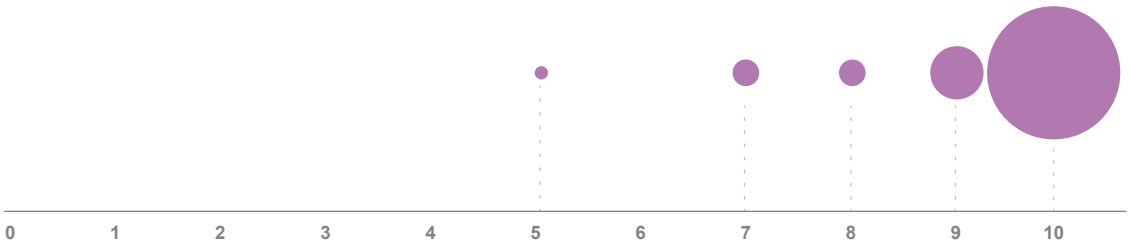

Figure 5.18: Number of scores for *Flexibility and efficiency of use*

This heuristic has a very high percentage of users (around 84%) evaluate the system with a classification between *8* and *10*. More than half of the studies had a perfect score of *10*, which is due to the smoothness of the system (even with messaging systems and REST calls implemented and operational) and the ease of access to whatever desired page.

5.3.4.8 *Aesthetic and minimalist design*

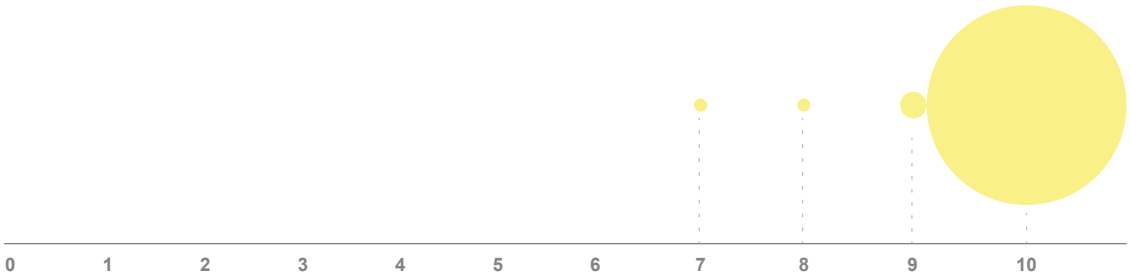

Figure 5.19: Number of scores for *Aesthetic and minimalist design*

Aesthetic and minimalist design was by far the highest-graded heuristic among all. Most of the users found the look of the system to be pleasing to the eye as well as the information presented in a transparent, non-overwhelming manner. The importance of this heuristic relies on how such a substantial amount of information (such as the Beach page, which is data-heavy) could be presented without confusing the users and most importantly having a clear, uniformized setting so that every user could easily and quickly take their conclusions from each page.

With 95% of evaluators rating the system on a *very* positive note ranged from *8* and *10*, and 79% grading a perfect score of *10*, this is by far the strongest heuristic of the system.

5.3.4.9 Help users recognize, diagnose, and recover from errors

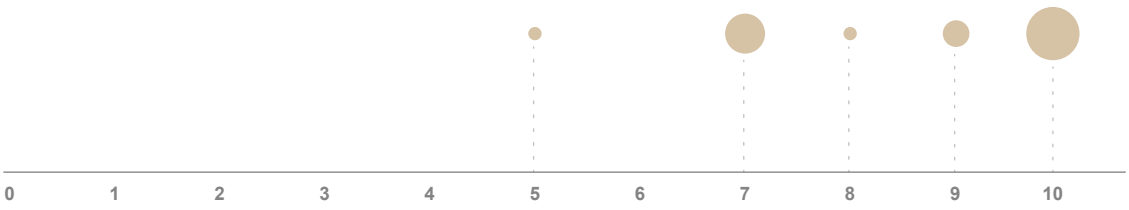

Figure 5.20: Number of scores for *Help users recognize, diagnose, and recover from errors*

The last heuristic evaluated was the hardest to classify for all evaluators, with only 58% of the users grading the system based on it. This was mostly due to the lack of error situations uncovered by the evaluators after testing the application.

Nevertheless, those that felt capable of rating the system on it mostly gave a positive note, with all scores ranging positively from *5* to *10*, with a majority of a perfect score of *10*.

For an overall look of the situation at hand, let's analyze the Radar chart of the Heuristic results in figure 5.21.

Figure 5.21: Heuristic evaluation: Radar chart

Following the same pattern of organization, it is possible to observe that the circles with the higher radius are all focused on the outer line of the radar chart. What this means is that the number of high scores for each heuristic is more significant than lower scores: an emptied center is a **positive** factor, as it suggests that the number of low scores is low or even non-existent.

Each color, as the legend of the picture suggests, implies a different heuristic.

5.3.5 Final Usability test conclusions

The tests performed had some factors to which the evaluators were subjected. The fact that each action was timed and compared to the time they thought it had taken them to do it was a great indicator of how easy or hard the application felt to them. The main feeling retrieved from the evaluators was that the application was most appealing to the eye, the pages were vibrant with information, but not too much to overwhelm the typical users. Iconography was amiss, due to the particularity that the system as a whole had two applications that had a similar theme, but different functionalities.

Although some mix-ups were made when it came to selecting the correct icon, it was swift, and intuitive to the user, as the icon to the Delivery page easily translated into delivery "snack-food" type of product.

Markers on some pages like the parking were also not very perceptible, as it was not very clear what they represented. Suggestions such as using different markers with the proper signalization of what they represented were the referred ones.

Other features that weren't very perceptible was, for example, the "Estado actual:" feature in the Parking page. When it was supposed to represent the current state of the Parking space as a whole (i.e., Green signaling when it contains many spaces in comparison to the total number of sensors distributed across the beach area, Yellow/Orange for medium availability and Red for very few availabilities), evaluators often confused this signal as one that signaled the current state of sensors (online/offline).

CHAPTER 6

Conclusions

After setting the foundation behind the theme of this dissertation, building the system, approaching multiple solutions, and even testing these on several ordinary and even people who rely heavily on technology, it is possible to conclude that smart city applications can reach heights that for some reason, are being hindered and not reaching the full potential that they're destined to.

Smart city applications have an incredible array of resources, dedicated components (i.e., sensors, technological services, data gathering methods, etc.), and applicable targets. Investment is still something that has been in the works, and it is the main reason why these developments do not move forward as fast as they could.

Since the number of IT people is increasing every day, it's possible to see use-case applications (such as the ones discussed in this dissertation) a lot more, which ends up spiking interest in companies and even institutional organizations or city hall administrators that have the power to put this type of projects into motion.

6.1 Outcomes of the developed work

After building a functional application that has been tested by some different users and evaluated as such, it is possible to conclude that the system itself, while in need of some refinement, is a good stepping stone for other applications to come around and be influenced by it. It was proven that the application is indeed *useful* among the evaluators, as well as an excellent example for smart city applications. Most of the evaluators inquired responded to *Do you see yourself using this application in the future?* with a confident, positive note - the remaining people that were questioned and answered negatively, did so due to the application being focused on an area out of bounds for them. This proves that this type of application is indeed sought and most of all, useful for the typical citizen, the next point in action being the expanding of the area covered, as well as the type of applications provided.

There's a significant volume of work that could be furthered in this system's implementation. It is safe to assume that a system with a vast spectrum like this can evolve and expand

significantly as the user's needs also grow. With the advance of technologies and the number of new devices and components that appear every day, it is possible to produce use cases such as the ones showcased here but with different data sources and appliances.

Since this system is built as a SPA, it is also capable of hosting more single applications with their functionalities much like the Parking page, Delivery, and all the others. Pages involving public transports, waste management, and other smart solutions depicted in the previous chapters can also be one of the following projects implemented and hosted on the system. The future work and evolution of this project heavily rely on this, as it can be built and further enhanced by additional features to the existent pages and possibly more single pages.

6.2 Future work

Seeing as this project is directly correlated to PASMO, what is being projected here is an example of a use case application that uses the resources developed and maintained by platforms such as SCoT, the hardware developed in the PASMO project, and all technologies that the project itself can provide.

That being said, the possibilities of building different applications and different outcomes are endless, and developers strive to continue to create and build smart technologies and smart applications. The resources that are at hand are meant to be used in a useful way, as well as experiment with new ideas and even new information systems.

The creation of smart city applications is one that is being received in a very positive note by population overall, and the development and research surrounding it, as it grows every day in numbers, should not cease as more resources are provided, more positive feedback, and more future investments. The trend should be to follow this calling and create, develop, research, and invest more.

6.2.1 Future work on existent applications

Looking at the future work that can be applied to the already existent structures and application that exist, it is better to organize these ideas into sections taking into account multiple factors such as:

- **Usability test results:** It is interesting to analyze the gathered feedback of the numerous opinions provided by all the evaluators; these, in particular, are considered the best source for gathering future work suggestions.
- **Self-introspective:** Of course as a developer, some features and capabilities were unfortunately disregarded due to a time limit in implementation. It is a fun exercise to look into all the ideas produced initially (over 20 ideas were considered, and it fits the context to take some of the best ones and discuss them in this chapter).

6.2.1.1 Parking Page

The Parking page is one of the most potent and information-heavy pages available for the system as a whole. It is highly regarded (among the evaluators, mostly) as the most helpful

and innovative application.

The main issue that the evaluators found with this page is that it was limited to features that could be covered and already are by other systems such as Google Maps and Waze. These applications are complicated on their own and are implemented with their single purpose: To provide navigation and GPS features with auxiliary and community-sourced information (Waze, for example, is highly dependent on community involvement for their development and growth).

The parking page is an excellent example of a vast source of untapped data that could evolve with the number of users providing their data and contributions. For example, instead of assuming based on previous data, users would be able to inform the system of the current conditions of the area regarding occupancy or traffic.

Other exciting features that could be applied here would be to set a history accessible to each user with the information of the last parking spaces they've parked at (provided that the user himself uses the *Save car location* feature), better suggestions of parking spaces according to previous choices from the user, signaling of recent accident events or irregularities on traffic happening[1].

The most exciting features to implement on this page are the ones that will actually have an impact on the user's usability and ultimately help them and improve the quality of the service provided. This is achievable by offering the user better choices, routes, and the total freedom to chose what to do even with so much available to them.

6.2.1.2 Beach Page

There is a clear goal for the utilization and implementation of the beach page that is to inform and offer knowledge on the variation of values from different ranges of time.

There's an opportunity here to create a simple feature that would be able to call a lifeguard to the person calling's location. That would also imply, however, for the lifeguards to have a subscribing service of their own to receive the notification of those users correctly- that is a possible future work implementation.

Another idea that had been discussed previously was to implement a people counter[2], which can manifest itself in a multitude of ways, being able to *count* people from different technological points of views:

- **Video counters:** Video counters are set with an algorithm that is capable of mapping the different people in a specific area they are filming. It is capable of being seamlessly applied in areas with high altitude that would be capable of a wider range of coverage.
- **Wi-fi counting:** Using a Wi-fi receiver to pick up unique Wi-fi management frames emitted by different smart devices (such as tablets or smartphones) within range [49], it's possible to have an estimative of the number of people in the area (this is highly jeopardized by the number of devices each person has connected to the Wi-fi, and it assumes that each person actually even has a smart device).

[1] An idea based on this has been actually implemented as part of another project from IT, which uses Vehicular ad hoc communications to communication from vehicle to vehicle

[2] A people counter is an electronic device used most frequently in retail stores and indoor locations

These are only a few practical examples of how it would be possible to implement this type of feature on the Beach page. After selecting the technology to gather this type of data, the implementation could take place on the Client-side and present the information in the style of a choropleth map, that could be activated on user-demand. This would also be considered as a factor on the user's criteria of choice when selecting an area to perform beach activities- together with the temperature, wind, and all other sensor-retrieved data.

6.2.1.3 Delivery

For the Delivery section, most of the evaluators specifically suggested one feature: a time-based prediction on how much time the person ordering would have to wait for the Deliverer to reach them.

This implementation, however, would have to take into account a multitude of factors, as due to the nature of this system in particular, this implementation was proved to contain some more hardships than it was predicted in the beginning.

If more than one user has requested for the same deliverer- which the system accepts and, in practice, is something that usually happens, the deliverer partakes in an optional situation. Ideally, the Deliverer would go to the first person that has requested him, and then attend to the next person. Such a case is impractical if the Deliverer is on-foot. It would be too time-consuming to traverse the length of the route from the current Deliverer's position to the first person that has requested him, seeing that he is much further than the second person that has requested him. For calculating the remaining time, the Vehicular Routing Problem (VRP) of the subject at hand would have to be assessed.

Creating a solution for the VRP would mean to optimize all routes necessary within the spectrum of clients that the Deliverer has to attend to, and generating the best solution to satisfy all the stakeholder's needs.

Furthermore, after obtaining the optimal routes necessary for the Deliverer to perform, it would be relatively simple to apply a Time-window based on the speed of the Deliverer, and the distance he will have to cross.

While this issue was approached, it was not developed to the fullest due to lack of time.

References

[1] *Top Countries by Smartphone Penetration & Users | Newzoo*. [Online]. Available: https://newzoo.com/insights/rankings/top-50-countries-by-smartphone-penetration-and-users/ (visited on 05/02/2018).

[2] I. .-.-. I. de Telecomunicações, *It - website*. [Online]. Available: https://www.it.pt/.

[3] J. Ferreira, J. Fonseca, D. Gomes, J. Barraca, B. Fernandes, J. Rufino, J. Almeida, and R. Aguiar, "Pasmo: An open living lab for cooperative its and smart regions", in *2017 International Smart Cities Conference (ISC2)*, Sep. 2017, pp. 1–6. DOI: 10.1109/ISC2.2017.8090866.

[4] E. Commission, *Data protection | european commission*. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection_en.

[5] G. News and M. Limited, *Smart cities: Are you willing to trade privacy for efficiency? | news | the guardian*. [Online]. Available: https://www.theguardian.com/news/2014/apr/04/if-smart-cities-dont-think-about-privacy-citizens-will-refuse-to-accept-change-says-cisco-chief.

[6] N. Komninos and E. Sefertzi, "Intelligent cities: R&d offshoring, web 2.0 product development and globalization of innovation systems", Jan. 2009.

[7] L. Anthopoulos and P. Fitsilis, "From online to ubiquitous cities: The technical transformation of virtual communities", in *Next Generation Society. Technological and Legal Issues*, A. B. Sideridis and C. Z. Patrikakis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 360–372, ISBN: 978-3-642-11631-5.

[8] M. Jang and S.-T. Suh, "U-city: New trends of urban planning in korea based on pervasive and ubiquitous geotechnology and geoinformation", in *Computational Science and Its Applications – ICCSA 2010*, D. Taniar, O. Gervasi, B. Murgante, E. Pardede, and B. O. Apduhan, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 262–270, ISBN: 978-3-642-12156-2.

[9] B. Felten, "Stokab Helps Build a Smart Stockholm", Diffraction Anlysis, Stockholm, Tech. Rep., 2015, p. 22. [Online]. Available: https://www.stokab.se/Documents/Nyheter%20bilagor/Stokab%20helps%20build%20a%20smarter%20Stockholm.pdf.

[10] M. A. Moser, "What is smart about the smart communities movement?", [Online]. Available: www.ucalgary.ca.

[11] E. Labs. (). Smart bins & level sensors | case studies | ecube labs, [Online]. Available: https://kr.ecubelabs.com/case-studies/. (access: 26.04.2018).

[12] C. of Chicago. (). Chicago smart lighting program, [Online]. Available: http://chicagosmartlighting-chicago.opendata.arcgis.com/. (access: 26.04.2018).

[13] Tvilight. (), [Online]. Available: https://www.tvilight.com/. (access: 26.04.2018).

[14] H. Peng, Z. Bohong, and K. Qinpei, "Smart City Environmental Pollution Prevention and Control Design Based on Internet of Things", *IOP Conference Series: Earth and Environmental Science*, vol. 94, no. 1, p. 012 174, Nov. 2017, ISSN: 1755-1307. DOI: 10.1088/1755-1315/94/1/012174. [Online]. Available: http://stacks.iop.org/1755-1315/94/i=1/a=012174?key=crossref.4fa6a9b24c985ee8a36627012fc53894.

[15] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies", *IEEE*

Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2456–2501, 2017, ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2736886. [Online]. Available: http://ieeexplore.ieee.org/document/8003273/.

[16] C. C. S. Department. (). Cmu scs coke machine, [Online]. Available: http://www.cs.cmu.edu/~coke/. (accessed: 24.04.2018).

[17] (). SenseMyCity, [Online]. Available: https://sensemycity.up.pt/ (visited on 04/30/2018).

[18] (). Ínicio - águeda, [Online]. Available: http://agueda.isasmartcity.com/?Locale=pt_PT (visited on 04/30/2019).

[19] *SmartSantander*. [Online]. Available: http://www.smartsantander.eu/ (visited on 05/07/2018).

[20] B. C. Hall. (). Mobility | barcelona city council, [Online]. Available: http://mobilitat.ajuntament.barcelona.cat/en/. (access: 30.04.2018).

[21] IBM. (). Ibm news room - 2009-10-01 ibm launches new advanced analytics center in new york - united states, [Online]. Available: https://www-03.ibm.com/press/us/en/pressrelease/28512.wss. (access: 30.04.2018).

[22] C. Ibsg, J. Frazier, and T. Touchet, "Cisco Internet Business Solutions Group (IBSG) Transforming the City of New York New Platform for Public-Private Cooperation Ushers in Smart Cities of the Future", [Online]. Available: https://www.cisco.com/c/dam/en%7B%5C_%7Dus/about/ac79/docs/ps/motm/City-24x7%7B%5C_%7DPoV.pdf.

[23] Vodafone, *Smart cities*. [Online]. Available: https://negocios.vodafone.pt/smart-cities/ (visited on 06/04/2018).

[24] (). Lighting living lab, [Online]. Available: http://www.lighting-living-lab.pt/.

[25] (). Rener living lab, [Online]. Available: http://www.inteli.pt/pt/go/mobielivinglab.

[26] (). Smart rural living lab, [Online]. Available: http://www.cm-penela.pt/artigos-174.

[27] Fiware, *Developers | FIWARE*. [Online]. Available: https://www.fiware.org/developers/ (visited on 05/10/2018).

[28] ——, *Quick fiware tour guide*. [Online]. Available: http://fiwaretourguide.readthedocs.io/en/latest/real-time-processing-of-media-streams/introduction/ (visited on 05/10/2018).

[29] M. Antunes, J. P. Barraca, D. Gomes, P. Oliveira, and R. L. Aguiar, "Smart Cloud of Things: An Evolved IoT Platform for Telco Providers", *Journal of Ambient Wireless Communications and Smart Environments (AMBIENTCOM)*, vol. 1, no. 1, pp. 1–24, Jul. 2016, ISSN: 2246-3410. DOI: 10.13052/ambientcom2246-3410.111. [Online]. Available: http://www.riverpublishers.com/journal%7B%5C_%7Dread%7B%5C_%7Dhtml%7B%5C_%7Darticle.php?j=AMBIENTCOM/1/1/1.

[30] SOA Work Group, *Service-Oriented Architecture Standards | The Open Group*. [Online]. Available: http://www.opengroup.org/standards/soa (visited on 05/28/2018).

[31] Jakob Nielsen, "10 Usability Heuristics for User Interface Design", 1995. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/.

[32] P. W. Jordan, "Human factors for pleasure in product use", *Applied Ergonomics*, vol. 29, no. 1, pp. 25–33, Feb. 1998, ISSN: 0003-6870. DOI: 10.1016/S0003-6870(97)00022-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003687097000227?via%7B%5C%%7D3Dihub.

[33] S. Kujala, V. Roto, K. Väänänen-Vainio-Mattila, E. Karapanos, and A. Sinnelä, "UX Curve: A method for evaluating long-term user experience", *Interacting with Computers*, vol. 23, no. 5, pp. 473–483, Sep. 2011, ISSN: 09535438. DOI: 10.1016/j.intcom.2011.06.005. [Online]. Available: https://academic.oup.com/iwc/article-lookup/doi/10.1016/j.intcom.2011.06.005.

[34] Google, *Design - material design*. [Online]. Available: https://material.io/design/.

[35] Vodafone. [Online]. Available: http://praiaemdirecto.com:8080/praia2014/main.html (visited on 05/17/2018).

[36] ——, [Online]. Available: https://www.vodafone.pt/main/particulares/apps-servicos/praia-em-directo.html (visited on 05/17/2018).

[37] G. Hohpe, *Hub and Spoke [or] Zen and the Art of Message Broker Maintenance - Enterprise Integration Patterns*, 2003. [Online]. Available: http://www.enterpriseintegrationpatterns.com/ramblings/03%7B%5C_%7Dhubandspoke.html (visited on 05/08/2018).

[38] (). Mqtt, [Online]. Available: http://mqtt.org/ (visited on 05/08/2018).

[39] O. O. 2015. (). Mqtt version 3.1.1, [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (visited on 05/09/2018).

[40] Google. [Online]. Available: https://developers.google.com/identity/sign-in/web/ (visited on 05/21/2018).

[41] V. Agafonkin. [Online]. Available: https://leafletjs.com/ (visited on 05/15/2018).

[42] F. Richard, M. Tom, F. John, B.-B. Saman, and B. Ansis, *Id (software)*. [Online]. Available: https://github.com/openstreetmap/iD (visited on 05/15/2018).

[43] O. Community, *State of the map*. [Online]. Available: https://stateofthemap.org/ (visited on 05/15/2018).

[44] P. Team. [Online]. Available: http://flask.pocoo.org/docs/1.0/ (visited on 06/03/2018).

[45] A. Developer and M. Stern, *Essential Design Principles - WWDC 2017*, 2017. [Online]. Available: https://developer.apple.com/videos/play/wwdc2017/802/.

[46] IBM, *Ibm design research*. [Online]. Available: https://www.ibm.com/design/research/.

[47] ——, *Ibm design language*. [Online]. Available: https://www.ibm.com/design/language/.

[48] ——, *Ibm design thinking*. [Online]. Available: https://www.ibm.com/design/thinking/.

[49] S. Depatla, A. Muralidharan, and Y. Mostofi, "Occupancy Estimation Using Only WiFi Power Measurements", *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1381–1393, Jul. 2015, ISSN: 0733-8716. DOI: 10.1109/JSAC.2015.2430272. [Online]. Available: http://ieeexplore.ieee.org/document/7102673/.